Tampereen yliopisto

Juho Latvio

ISA-95-LIITÄNNÄISSPESIFIKAATION MUKAINEN OPC UA -PALVELIN

Teknisten tieteiden tiedekunta
Kandidaatintyö
Lokakuu 2019

TIIVISTELMÄ

Juho Latvio: ISA-95-LIITÄNNÄISSPESIFIKAATION MUKAINEN OPC UA -PALVELIN
Kandidaatintyö
Tampereen yliopisto
Automaatiotekniikka
Lokakuu 2019

Tämän opinnäytetyön tarkoitus on kuvata, miten OPC Unified Architecture ISA-95-liitännäisspesifikaation mukainen palvelin toteutetaan ja miten tämän liitännäisspesifikaation mukaisen palvelimen osoiteavaruus rakennetaan olemassa olevan OPC UA -tietomallin pohjalta.

Opinnäytetyössä tutustutaan OPC UA -protokollaan, ISA-95-standardiin ja OPC UA ISA-95-liitännäisspesifikaatioon. Protokollien ja standardien esittelyssä keskitytään esittelemään niihin liittyvät tietomallinnuksen käytännöt. Työssä kuvataan, miten olemassa olevan OPC UA -palvelimen osoiteavaruuden sisältö on kuvattu OPC UA ISA-95-liitännäisspesifikaation mukaisen OPC UA -palvelimen osoiteavaruuteen. Kandidaatintyössä esitetään OPC UA ISA-95-liitännäisspesifikaation mukaisen OPC UA -palvelimen toteutus käyttäen avoimen lähdekoodin OPC UA -ohjelmistokirjastoja.

Työn tuloksena toteutettiin OPC UA ISA-95-liitännäisspesifikaation mukainen palvelin, jonka osoiteavaruuden rakenne perustuu olemassa olevan OPC UA -palvelimen osoiteavaruuteen. OPC UA ISA-95-liitännäisspesifikaation mukaisen palvelimen osoiteavaruuden tietomalli rakennettiin muuttamalla olemassa olevan OPC UA -palvelimen osoiteavaruuden sisältämän informaation rakenne ensin ISA-95-standardin mukaiseksi ja tämän jälkeen muuttamalla se vielä OPC UA ISA-95-liitännäisspesifikaatiossa kuvattujen tietomallien kanssa yhteensopivaksi. OPC UA ISA-95-liitännäisspesifikaation mukainen informaatiomalli toteutettiin OPC UA ISA-95-palvelimen osoiteavaruuteen käytettyjen ohjelmistokirjastojen dokumentaation sekä ohjelmistokirjastojen lähdekooditiedostojen tarjoaman tiedon perusteella. ISA-95-standardin mukaisen tiedon esitystavan ansiosta palvelinta voidaan hyödyntää esimerkiksi tuotantoprosessien integroinnissa tehtaan tuotannonohjaus- tai ERP-järjestelmän kanssa.

Avainsanat: OPC UA, ISA-95, OPC UA ISA-95-liitännäisspesifikaatio

SISÄLLYSLUETTELO

KUVALUETTELO

OHJELMA- JA ALGORITMILUETTELO

LYHENTEET JA MERKINNÄT

B2MML	Business To Manufacturing Markup Language
ERP	Enterprise Resource Planning
OPC UA	OPC Unified Architecture
PFC-kaavio	Procedural Function Chart -kaavio
PI-kaavio	Putkisto- ja instrumentointikaavio
XML	Extensible Markup Language

1 JOHDANTO

Standardit ovat määrittelyitä, joiden avulla kuvataan tiettyyn asiakokonaisuuteen liittyvät säännöt, ohjeistukset sekä toteuttamiseen liittyvät tavat. Standardeja käyttäen voidaan saada aikaan esimerkiksi tuotteiden ja palveluiden parempi yhteensopivuus [4][15]. Myös automaatioalalla on useita standardeja niin turvallisuuteen, tiedonsiirtomenetelmiin kuin tiedonkuvaukseen liittyen. Tässä opinnäytetyössä tutustutaan teollisuudessa keskeisiin OPC Unified Architecture ja ISA-95-standardeihin.

OPC UA on tiedonsiirtostandardi, joka on kasvattanut suosiotaan teollisuudessa viime vuosina. OPC UA -protokollan suosiota teollisuudessa selittää muun muassa sen hyvä skaalautuvuus, alustariippumattomuus sekä tietoturvallisuus [6].

ISA-95-standardin tarkoitus on määrittää toiminnanohjausjärjestelmän ja tuotannonohjausjärjestelmän välillä siirrettävän tiedon kuvaustapa. Siirrettävän tiedon standardin mukaisella kuvauksella pyritään parantamaan esimerkiksi tuotantolaitoksen eri operaatiokerrosten välisen tiedonsiirron tehokkuutta sekä tiedon hallintaa [1, s. 9–10][14, s. 7].

Kandidaatintyössä selvitetään, miten OPC UA ISA-95-liitännäisspesifikaation mukainen palvelin toteutetaan ja miten OPC UA ISA-95-liitännäisspesifikaation mukaisen palvelimen osoiteavaruus rakennetaan olemassa olevan OPC UA -tietomallin pohjalta. Tutkimuskysymykseen etsitään vastaus tutustumalla sekä OPC UA -tiedonsiirtostandardiin että ISA-95-tietomalliin. Työssä tutustutaan lisäksi OPC UA ISA-95-liitännäisspesifikaatioon ja sen määrittämään tietomalliin.

Työssä implementoidaan OPC UA -palvelin, joka toteuttaa OPC UA ISA-95-liitännäisspesifikaation mukaisen tietomallin. Työssä toteutettava palvelin muuttaa sellunkeittosimulaattoriprosessia ohjaavan OPC UA -palvelimen OPC UA -tietomallin ISA-95-tietomallin mukaiseksi. Toteutettava palvelin tarjoaa rajapinnan, jonka avulla minipanosprosessia voidaan ohjata, ja prosessista voidaan lukea laitteiden mittausarvoja.

Opinnäytetyössä ensimmäisenä tutustutaan keskeisiin protokolliin ja standardeihin, jonka jälkeen kuvataan, miten OPC UA ISA-95-liitännäisspesifikaation mukainen palvelin voidaan toteuttaa käyttäen avoimen lähdekoodin ohjelmistokirjastoja. Opinnäytetyössä kuvataan sekä keskeisimmät kohdat palvelimen toteutuksesta että palvelimen sisältämän osoiteavaruuden rakenne. Lopuksi esitellään työn tulokset.

2 TEOREETTINEN TAUSTA JA LÄHTÖKOHDAT

2.1 OPC UA

Koska OPC UA -protokolla on laaja kokonaisuus sisältäen määrittelyn sekä tiedonsiirtoon että tietomallinukseen [5, s. 10], ei opinnäytetyössä voida selittää protokollan kaikkia ominaisuuksia. Opinnäytetyön kannalta keskeistä ei ole OPC UA -protokollan tiedonsiirtoon liittyvät menetelmät, vaan protokollaan liittyvä tietomallinnus, joten tämän vuoksi seuraavissa alaluvuissa on keskitytty OPC UA -protokollan perusteisiin sekä tietomallinnuksen periaatteisiin.

2.1.1 Yleistä

OPC UA on teollisuuteen tarkoitettu tiedonsiirtoprotokolla, joka tarjoaa alustariippumattoman ja turvallisen tavan siirtää tietoa eri järjestelmien välillä. OPC UA on kehitetty Classic OPC:n seuraajaksi pyrkien tarjoamaan vastaavan toiminnallisuuden kuin Classic OPC, kuitenkin laajentaen protokollan ominaisuuksia vastaamaan nykyajan vaatimuksia [5, s. 8–9]. Classic OPC on OPC Foundation -säätiön määrittämä protokolla, joka perustuu Microsoftin COM ja DCOM teknologioihin. Classic OPC määrittää kolme keskeistä rajapintaa, jotka ovat Data Access, Alarms & Events ja Historical Data Access. Data Access -rajapinta määrittää toiminnallisuuden, jonka avulla voidaan käsitellä OPC-palvelimen sisältämää prosessidataa. Alarms & Events -rajapinta tarjoaa tavan, jonka avulla voidaan käsitellä tapahtumiin liittyvää tietoa. Historical Data Access -rajapintaa voidaan käyttää historiatiedon käsittelemiseen [5, s. 3–4].

OPC UA -protokollassa keskeisiä kokonaisuuksia ovat tietomallinnus sekä tiedonsiirtomenetelmät. Protokollassa esitetään tietomallinnuksen osalta yleiset tavat ja säännöt OPC UA -tietomallien luontiin. Tiedonsiirtomenetelmiin liittyen OPC UA -standardissa kuvataan esimerkiksi käytettävät tiedonsiirtoprotokollat ja niihin liittyvät tietoturvaominaisuudet [5, s. 10].

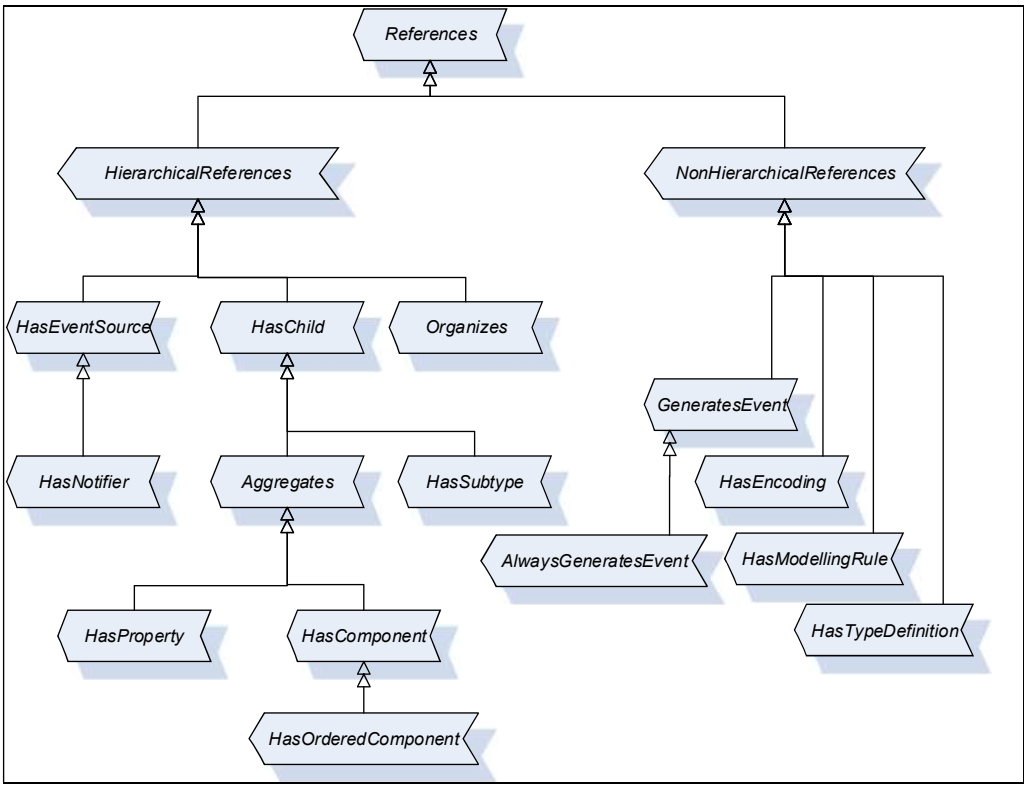
2.1.2 OPC UA -tietomalli

OPC UA -tietomallilla tarkoitetaan kokoelmaa standardin mukaisia solmu- ja viiteinstansseja. Tietomalliin liittyy keskeisesti myös standardin mukaiset solmujen tietotyypit [6]. Tietomallin avulla muodostetaan OPC UA -palvelimen osoiteavaruus. Osoiteavaruudella tarkoitetaan palvelimen sisältävää tietoa, joka on OPC UA -palvelinta käyttävän asiakassovelluksen hyödynnettävissä [9, s. 2]. Osoiteavaruuden sisältämän tiedon on tarkoitus olla kuvaus OPC UA -palvelimeen liittyvän fyysisen systeemin tietomallista. Osoiteavaruuden solmut esittävätkin fyysisen systeemin olioita, niiden määrityksiä sekä olioiden välisiä viitteitä [9, s. 15]. Osoiteavaruuteen voidaan mallintaa esimerkiksi fyysiseen järjestelmään liittyvät laitteet, niihin liittyvä tieto sekä eri laitteiden väliset suhteet.

OPC UA -tietomallinnuksessa hyödynnetään oliomallinnuksen keinoja. Esimerkiksi informaatiomallinnuksessa keskeisiä asioita ovat olioiden tyyppihierarkiat sekä periytyminen [5, s. 19]. Tietomallinnuksen perusteella muodostettu osoiteavaruus on rakenteeltaan hierarkkinen. Hierakkisuus näkyy OPC UA -palvelimissa esimerkiksi palvelimien osoiteavaruuden standardin mukaisella ylätasolla. Toisaalta, koska osoiteavaruuden solmut voivat viitata toisiin solmuihin, osoiteavaruuden hierarkkisuus voi rikkoontua [9, s. 12]. OPC UA -osoiteavaruuden rakenne ei ole protokollan puolesta rajattu, joten OPC UA -palvelimen osoiteavaruuden sisältämistä solmuista voidaankin näin ollen rakentaa halutunlainen kokonaisuus [5, s. 20][9, s. 15].

OPC UA -tietomallit koostuvat solmuista ja solmujen välisistä viitteistä [5, s. 22]. OPC UA -osoiteavaruus rakentuu erilaisista solmuista. Solmu on mallinnuksen peruskäsite, jolla kuvataan OPC UA -tietomalliin liittyviä eri komponentteja [9, s. 15]. Eri komponenteille on erilaisia luokkamäärittelyjä sen mukaan, mihin niitä käytetään [5, s. 22]. Solmu voi kuvata olioinstanssia eli osoiteavaruuden osaa, joka esittää systeemiin liittyvää fyysistä tai abstraktia osaa [9, s. 4], tai sen tarkoitus voi olla esimerkiksi tyyppimäärittelyn tarjoaminen [5, s. 22]. Jokaiseen solmuun liittyy joukko attribuutteja, jotka kuvaavat solmujen ominaisuuksia. Jokaiselle solmulle on määritetty joukko attribuutteja, jotka liittyvät kaikkiin osoiteavaruuden solmuihin. Jokaiseen solmuun liittyvien attribuuttien lisäksi eri solmuluokkiin (NodeClass) kuulu attribuuttien joukko, joka kuvaa tiettyä solmuluokkaa [5, s. 22].

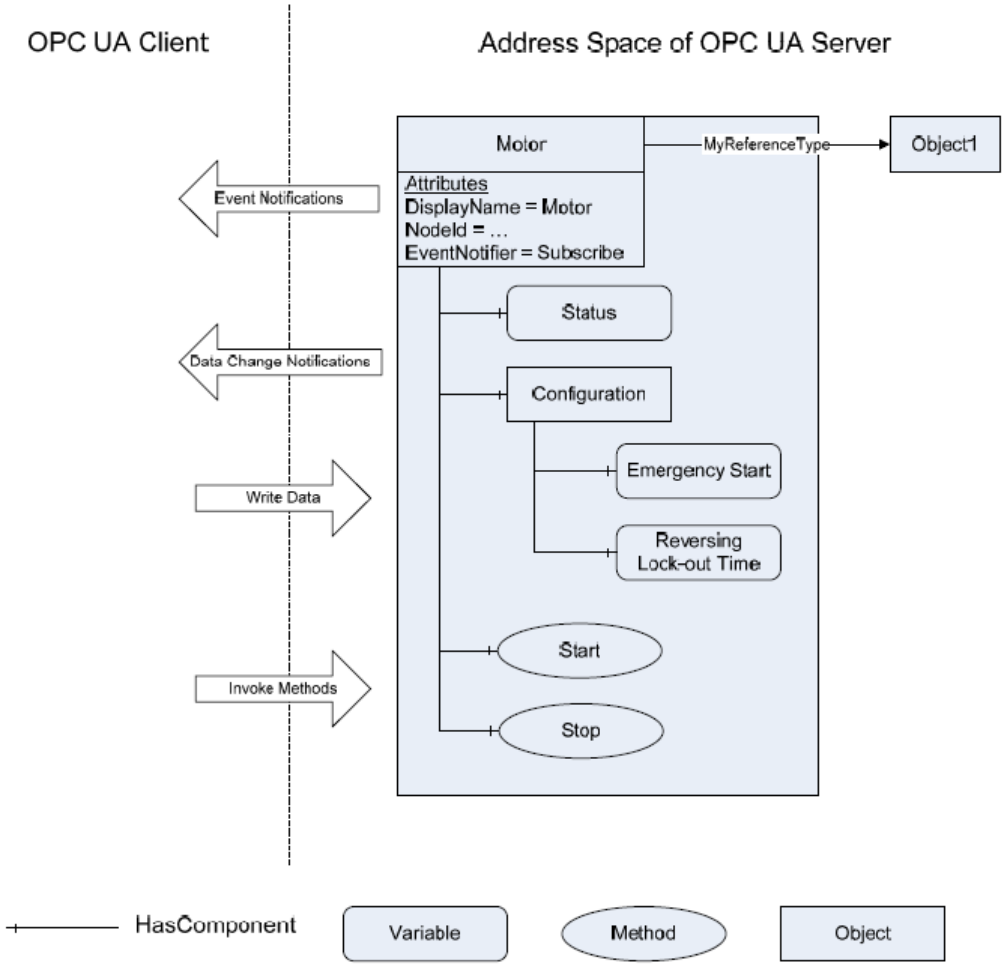
Kahden solmun välillä olevilla viitteillä on erilaisia merkityksiä OPC UA -tietomallissa. Erityyppisiä viitteitä voidaankin käyttää kuvaamaan OPC UA -osoiteavaruuden solmujen välistä suhdetta. Viitteet voidaan jakaa kahteen kategoriaan: hierarkkisiin ja ei-hierarkkisiin viitteisiin. Kun osoiteavaruuden solmuilla halutaan mallintaa hierarkkista rakennetta, hyödynnetään hierarkkisia viitteitä, ja kun solmujen välinen suhde ei ole hierarkkinen, käytetään ei-hierarkkista viitettä [5, s. 25–26]. Esimerkki ei-hierakkisesta viitteestä on HasTypeDefinition-viite, jonka avulla esimerkiksi olio voidaan liittää siihen kuuluvaan tyyppimäärittelysolmuun [10, s. 57]. HasSubType-viite on puolestaan esimerkki hierarkkisesta viitteestä, jonka tarkoitus on kuvata tietyn solmun periytymistä viittaavasta solmusta [10, s. 57]. Kuvassa 2.1 on esitettynä OPC UA -standardin viitehierarkia ja hierarkiaan kuuluvat eri viitteet.



Kuva 2.1. *OPC UA -viitteiden tyyppihierarkia [10, s. 55].*

OPC UA:n kannalta keskeisiä solmuluokkia ovat Object- ja Variable-solmuluokat [5, s. 30]. OPC UA -tietomallissa Variable-solmuluokkaa käytetään määrittämään Object-solmuluokan instansseja. OPC UA -tietomallinnuksessa Variable-tyyppisiä solmuja käytetään esittämään arvoja. Variable-tyyppisiin solmuihin voidaan esimerkiksi tallentaa jonkin anturin tuottama mittausarvo tai jokin järjestelmään liittyvä tiedosto [5, s. 30][10, s. 7]. Variable-solmut voidaan jakaa kahteen alityyppiin, joita ovat Properties ja DataVariables. Properties-solmuja käytetään kuvaamaan olioiden tunnusomaisia piirteitä. Oliossa erityyppisiä muuttujia voidaan käyttää esimerkiksi siten, että Object-solmun tiettyyn Data Variable -tyyppiseen muuttujaan tallennetaan anturista saatu mittausarvo ja Object-solmun yksittäiseen Property-solmuun tallennetaan mittauksen yksikkö [5, s. 58].

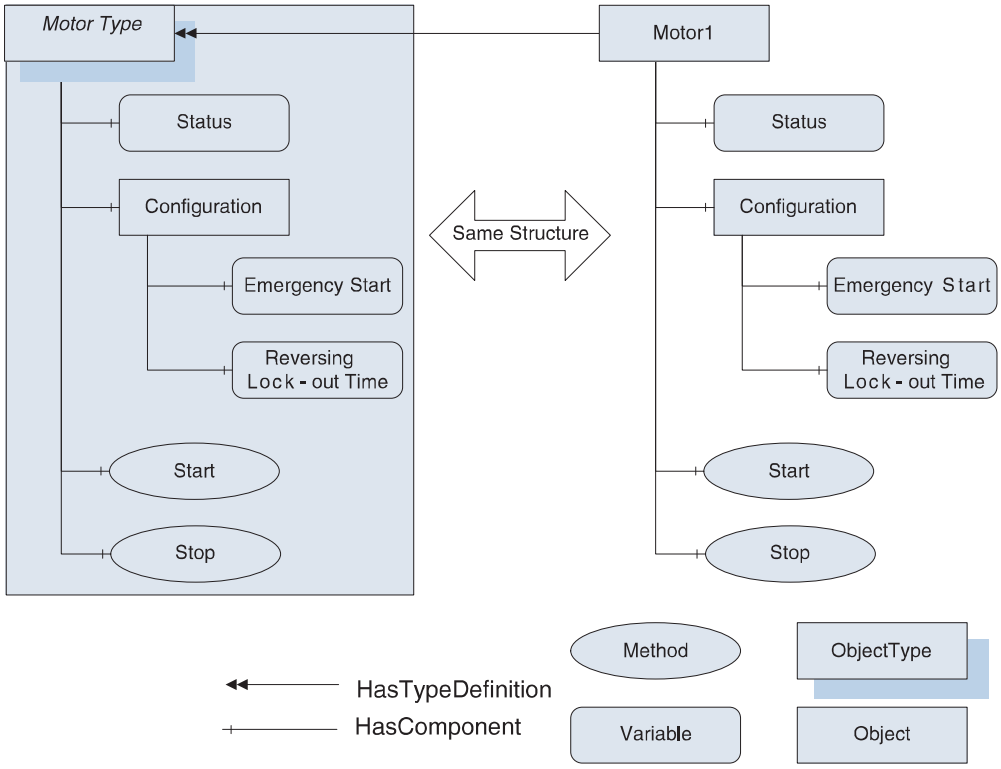
Object- ja Variable-solmuluokkien ohella keskeinen osa OPC UA -tietomallia on Method-solmuluokka [5, s. 30]. Metodit liittyvät Object-solmuihin ja niiden tarkoitus on tarjota OPC UA -palvelimeen liittyvälle asiakassovellukselle tapa vuorovaikuttaa palvelimen kanssa. Asiakassovellus voi kutsua jotain OPC UA -palvelimen metodia, minkä seurauksena palvelimessa suoritetaan jokin operaatio tai operaatioiden sarja. Metodien suorituksen päättyessä asiakassovellukselle palautetaan paluuarvo. Metodeille on määritetty tietyt sisääntuloparametrit, joiden avulla kutsuja voi siirtää palvelimelle metodin suorittamisessa tarvittavaa tietoa, sekä paluuarvot, joita kutsuja voi olettaa saavansa metodin suorituksen valmistuessa. Metodeja voidaan käyttää esimerkiksi OPC UA -palvelimen ohjaamaan prosessiin kuuluvan toimilaitteen käynnistämiseen [5, s. 30]. Kuvassa 2.2 on esitetty OPC UA -olio, johon liittyy sekä muuttujia että metodeja.



Kuva 2.2. *OPC UA -olio, johon liittyy muuttujia, metodeja sekä toisia olioita [5, s. 31].*

Olioille kuvataan tyyppimäärittelyt palvelimen osoiteavaruuteen. Tyyppimäärittelyt tehdään TypeDefinitionNode-solmuilla, jotka on liitetty OPC UA -instansseihin käyttäen HasTypeDefinition-viitettä. Olioinstanssien tyyppimäärittelyt ovat pakollinen osa OPC UA -osoiteavaruutta, mutta joissakin tilanteissa voidaan käyttää OPC UA:n määrittämiä tyyppimäärittelyiden kantatyyppejä. Kantatyyppejä voidaan käyttää esimerkiksi tilanteessa, jossa erikoistettuja tyyppimäärittelyitä ei ole tarpeen toteuttaa [10, s. 8]. Perusmäärittelyitä voidaan tarvittaessa laajentaa periyttämällä uusia tietotyyppejä standardin mukaisista perustietotyypeistä [5, s. 36].

OPC UA -osoiteavaruudessa olioiden tyypit voivat olla joko yksinkertaisia (Simple ObjectType) tai kompleksisia (Complex ObjectType). Kompleksisille olioiden tyyppimäärittelyille on ominaista se, että olion tyyppimäärittelyssä viitataan toisiin OPC UA -osoiteavaruuden solmuihin [5, s. 37]. Kuvassa 2.3 on esitetty kompleksinen olio. Kuvasta voidaan huomata, että tyyppimäärittelyssä Motor Type -olio viittaa Configuration-olioon ja näin ollen rakentuu toisesta oliosta. Kuvasta voidaan myös nähdä, että Motor1-olion, jota määritetään kuvassa vasemmalla olevalla tyyppimäärittelyllä, rakenne vastaa tyyppimäärittelyn rakennetta.



Kuva 2.3. *Esimerkki kompleksisesta olion tyyppimäärittelystä [5, s. 43].*

Esimerkki yksinkertaisesta ObjectType-tyyppisestä oliosta on FolderType, jota käytetään osoiteavaruudessa olevien solmujen organisointiin ja järjestämiseen. Yksinkertaisen ObjectType-tyyppisen olion tarkoitus on kuvata OPC UA -osoiteavaruuden solmujen merkitystä ja semantiikkaa [5, s. 37].

Myös muuttujat voivat olla yksinkertaisia tai kompleksisia OPC UA -osoiteavaruudessa [5, s. 39]. Kompleksiset muuttujat ovat samankaltaisia kuin kompleksiset oliot rakentuen erillisistä osoiteavaruuden solmuista [5, s. 39][5, s. 48]. Kuten yksinkertaisten ObjectType-tyyppisten olioiden tapauksessa, yksinkertainen VariableType määrittää muuttujan semantiikan [5, s. 39].

OPC UA Address Space Model eli OPC UA -metamalli koostuu solmuluokista (NodeClasses) ja niihin liittyvistä attribuuteista [5, s. 81]. Address Space Model määrittääkin solmuluokkien kantatyypin, josta kaikki muut OPC UA -osoiteavaruuden solmuluokat periytyvät [10, s. 15]. Keskeinen osa metamallia on myös solmujen välisten viitteiden tietotyypit eli ReferenceType-määrittelyt [10, s. 18].

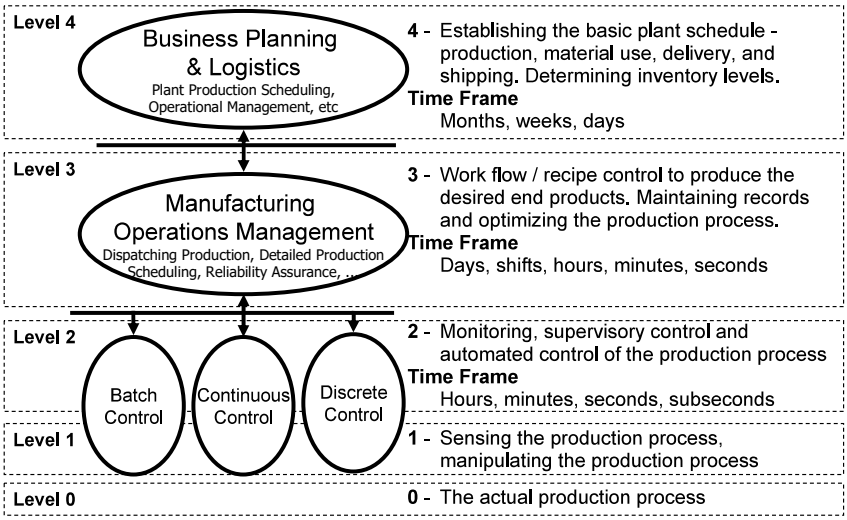
Informaatiomalli kuvaa OPC UA -palvelimen osoiteavaruuden rakennetta [9, s. 9]. Palvelimen osoiteavaruuteen liittyvät standardin mukaiset solmut määritetäänkin informaatiomallissa [11, s. 1]. Informaatiomallissa esitetään lisäksi käytettävien olioiden rakenne ja niihin liittyvät rajoitteet [5, s. 82]. OPC UA -informaatiomallit rakentuvat OPC UA -metamallin perusteella [5, s. 82][9, s. 3].

OPC UA -spesifikaatiossa on määritetty perustietomalli (Base Information Model). Joissain tapauksissa spesifikaation määrittämä tietomalli ei ole riittävä, joten tietomallia voidaan haluta laajentaa, jotta se sopisi paremmin käyttötilanteeseen. Mikäli tietomallia halutaan jatkaa, käytetään spesifikaation määrittämää perustietomallia uuden tietomallin pohjana [5, s. 19][5, s. 82].

2.2 ISA-95

ISA-95 on tiedonkuvausstandardi, jonka tarkoitus on määrittää tuotantolaitoksen eri kerrosten välillä siirrettävän tiedon rakenne [14, s. 6–7]. ISA-95-protokollan määrittämää tietomallia käytetään yleisesti tuotantolaitoksen kolmannen ja neljännen hierarkiakerroksen välisessä tiedonsiirrossa sekä kolmannella kerroksella kerroksen eri systeemien kesken tapahtuvassa tiedonvälityksessä [14, s. 9].

ISA-95-spesifikaatiossa tuotantolaitos jaetaan hierarkkisesti viiteen eri kerrokseen nollasta neljään. ISA-95-standardin hierarkiamallin neljännellä kerroksella tarkoitetaan tasoa, jonka vastuulla on hallita yrityksen toimintaa kokonaisuudessaan. Neljännen hierarkiakerroksen tehtäviä ovat esimerkiksi tuotantolaitoksen hajautettujen toimintojen aikatauluttaminen ja ohjaaminen sekä yrityksen varastojen ja logistiikan hallinta [1, s. 20]. Neljännellä tasolla on usein myös yrityksen ERP-järjestelmä [14, s. 8]. Tuotantolaitoksen kolmas kerros on vastuussa tuotantolaitoksen tuotannonohjauksesta. Kolmannen kerroksen toteuttamat toiminnallisuudet liittyvät esimerkiksi tuotantolaitoksen tuotannon aikatauluttamiseen, laadunvarmistukseen sekä tuotantolaitoksen sisäisen logistiikan hallintaan [14, s. 8]. Kerros kaksi huolehtii tuotantolaitoksen prosessien monitoroinnista ja ohjauksesta [1, s. 20]. Tasolla yksi tarkoitetaan antureita, jotka mittaavat fyysistä järjestelmää sekä toimilaitteita, jotka ovat osana ohjausjärjestelmää [14, s. 8]. Taso nolla käsittää itse fyysisen prosessin, jota ylemmiltä kerroksilta ohjataan. Kuvassa 2.4 on esitetty ISA-95-standardiin kuuluvat eri hierarkiakerrokset.

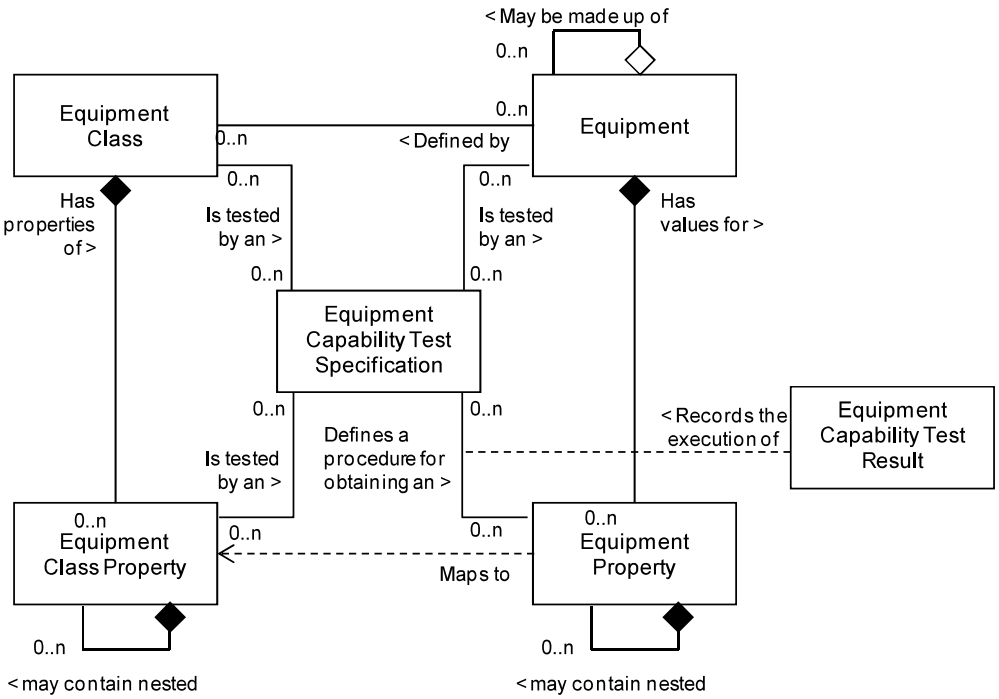


Kuva 2.4. *ISA-95-standardin määrittämä hierarkiamalli [1, s. 21].*

Esimerkkejä tuotantolaitoksessa siirrettävästä tiedosta ovat henkilöihin, laitteisiin ja materiaaleihin liittyvät tiedot. ISA-95 tarjoaakin erilaisia tietomalleja (Object Models), joita voidaan hyödyntää tiedon kuvauksessa eri tilanteissa. Standardiin on määritetty oliomalleja henkilöihin (Personnel information), laitteiden rooleihin (Role based equipment information), fyysisiin laitteisiin (Physical asset information), prosessin segmentteihin (Process segment information) sekä materiaaleihin (Material information) liittyvän tiedon kuvaamiseen [2, s. 27–46]. Opinnäytetyön kannalta keskeisiä oliomalleja ovat Role based equipment -oliomalli sekä fyysisiä laitteita kuvaava Physical asset -oliomalli.

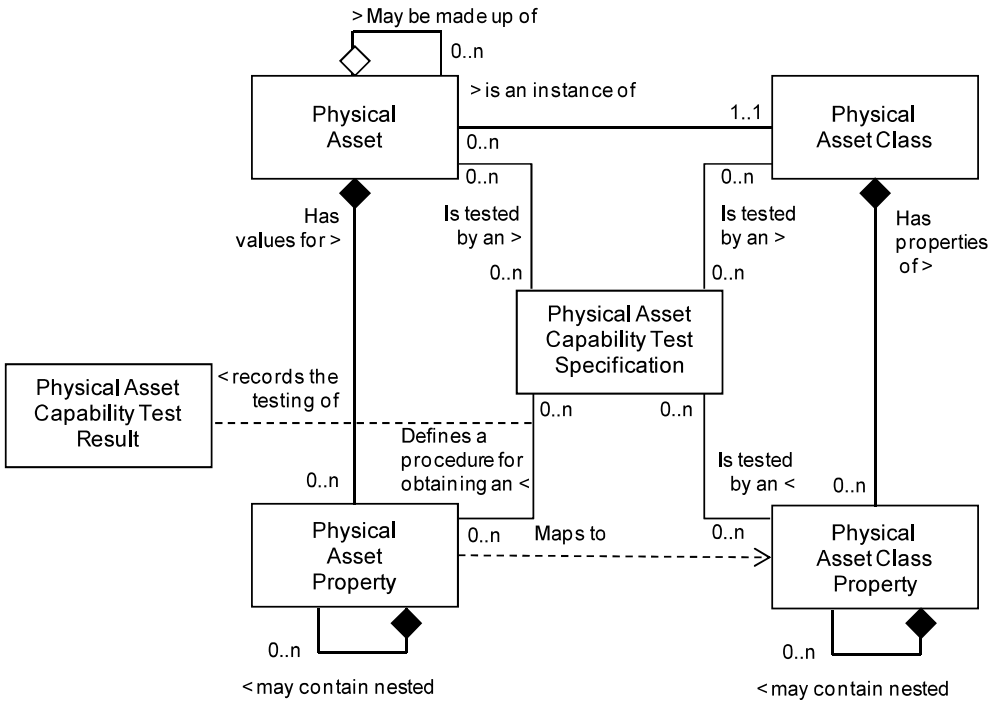
Role based equipment -oliomallia käytetään esittämään järjestelmän osien roolien määrittämät luokat ja niihin liittyvä tieto. Tällä tarkoitetaan esimerkiksi sitä, että prosessilaitteiston eri osat, kuten esimerkiksi pumput, venttiilit ja säiliöt, jaetaan omiksi loogisiksi kokonaisuuksikseen [14, s. 11]. Role based equipment -oliomalli ei ota kantaa järjestelmän fyysisiin laitteisiin, vaan se kuvaa vain loogisiin laitteisiin liittyvät luokat. Fyysisiin laitteisiin liittyvä tietomalli on kuvattu Physical asset -oliomallissa.

Kuvassa 2.5 on esitetty ISA-95-standardin määrittämä Role based equipment -oliomalli. Kyseisen oliomallin Equipment Class -luokka kuvaa järjestelmän eri osien luokkia. Yksi järjestelmään liittyvä Equipment Class -luokka voi kuvata esimerkiksi venttiiliä ja toinen Equipment Class -luokka voi määrittää esimerkiksi tankin. Equipment Class -luokkaan liittyy Equipment Class Property -luokka, jonka tarkoitus on tarjota Equipment Class -luokalle sen ominaiset parametrit [2, s. 33–35]. Equipment-luokalla määritetään tietty järjestelmän looginen laite. Equipment-luokka voi esittää esimerkiksi järjestelmän tietyn venttiilin loogisen informaatiomallin. Järjestelmään liittyvä venttiili voidaan nimetä esimerkiksi V123:ksi. Viitattaessa nimellä V123 tarkoitetaan tietyn venttiilin loogista tietomallia eikä itse fyysistä laitetta. Fyysisen laitteen erottaminen loogisesta kuvauksesta mahdollistaa esimerkiksi sen, että fyysinen laite, johon tietty looginen laite viittaa, voidaan vaihtaa huollon yhteydessä. Tällöin loogista laitetta ja siihen liittyvää tunnistetta ei tarvitse muuttaa, vaikka fyysinen laite vaihtuisikin [14, s. 54]. Equipment-luokkaan liittyy parametriluokka Equipment Property. Kyseinen luokka tarjoaa loogisen laitteen parametrit. Kyseiset parametrit voivat esiintyä itsenäisinä parametreina, mutta usein ne vastaavat Equipment Class Property -luokan instansseissa määritettyjä parametreja. Equipment Capability Test Specification -luokka kuvaa laitteelle ja sen parametreille suoritettavia testejä, joilla voidaan esimerkiksi määrittää laitteen soveltuvuus tiettyyn käyttötilanteeseen sekä määrittää laitteeseen liittyvien parametrien arvot. Testien perusteella saadaan tieto tuloksista, jotka esitetään Equipment Capability Test Result -luokassa [2, s. 36–38].



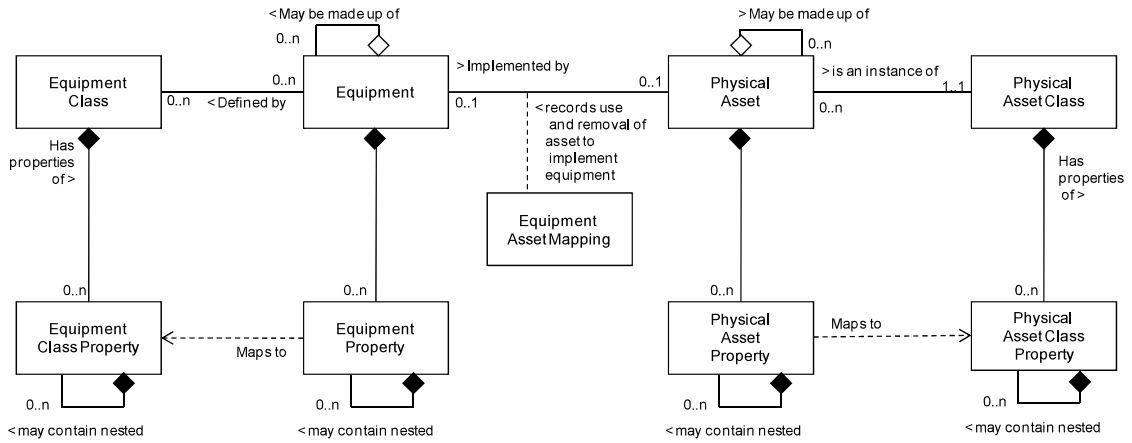
Kuva 2.5. *ISA-95-standardin määrittämä Role based equipment -oliomalli [2, s. 34].*

Physical asset -oliomallin, joka on esitetty kuvassa 2.6, tarkoitus on kuvata järjestelmään liittyviä fyysisiä laitteita. Järjestelmään liittyvä yksittäinen laite voi olla esimerkiksi järjestelmän fyysinen venttiili. Oliomallin Physical Asset -luokkaa käytetään määrittämään järjestelmän yksittäinen laite. Physical Asset -luokkaan voi liittyä Physical Asset Property -luokan instansseilla määritettäviä parametreja. Parametrit sisältävät tietoa laitteesta, kuten esimerkiksi laitteen valmistushetken. Physical Asset Class -luokkaa käytetään ryhmittelemään samankaltaisia laitteita omiksi ryhmikseen. Esimerkki laitteiden ryhmittelystä Physical Asset Class -luokkaa hyödyntäen on järjestelmän venttiilien jakaminen omiin ryhmiinsä perustuen laitteen valmistajaan ja malliin. Toisin sanoen samanmerkkiset ja -malliset venttiilit muodostavat oman venttiilien kokonaisuuden. Physical Asset Class Property -luokan instansseilla määritetään Physical Asset Class -luokan instansseille niitä määrittävät parametrit. Kuten Role based equipment -oliomalliin myös Physical asset -oliomalliin liittyy testejä, joilla laitetta voidaan testata ja joilla määritetään laitteeseen liittyviä Physical Asset Class Property ja Physical Asset Property -parametreja. Physical Asset Capability Test Specification -luokkaa käytetään kuvaamaan laitteisiin ja sen parametreihin liittyviä testejä. Physical Asset Capability Test Result -luokka kuvaa testeistä saatuja tuloksia [2, s. 40–45].



Kuva 2.6. *ISA-95-standardin määrittämä Physical asset -oliomalli [2, s. 41].*

Role based equipment -oliomallin ja Physical asset -oliomallin välillä on riippuvuussuhde, joka on esitetty kuvassa 2.7. Kuvasta voidaan huomata, että loogisen laitteen (Equipment-luokka) ja fyysisen laitteen (Physical Asset -luokka) välillä on riippuvuus. Loogisen laitteen ja fyysisen laitteen välinen suhde on väliaikainen, ja sitä kuvataan käyttäen Equipment Asset Mapping -luokkaa. Equipment Asset Mapping -luokka ylläpitää tietoa loogisen ja fyysisen laitteen välisen suhteen alkamisesta ja loppumisesta [2, s. 41][2, s. 46].



Kuva 2.7. *Role based equipment -oliomallin ja Physical asset -oliomallin välinen riippuvuus [2, s. 41].*

ISA-95-spesifikaatio ei tarjoa minkäänlaista toteutusta tiedonsiirtomallista. Abstraktin tietomallin perustan päälle on kuitenkin rakennettu konkreettisia toteutuksia, kuten esimerkiksi Business To Manufacturing Markup Language [14, s. 13]. B2MML on XML-pohjainen

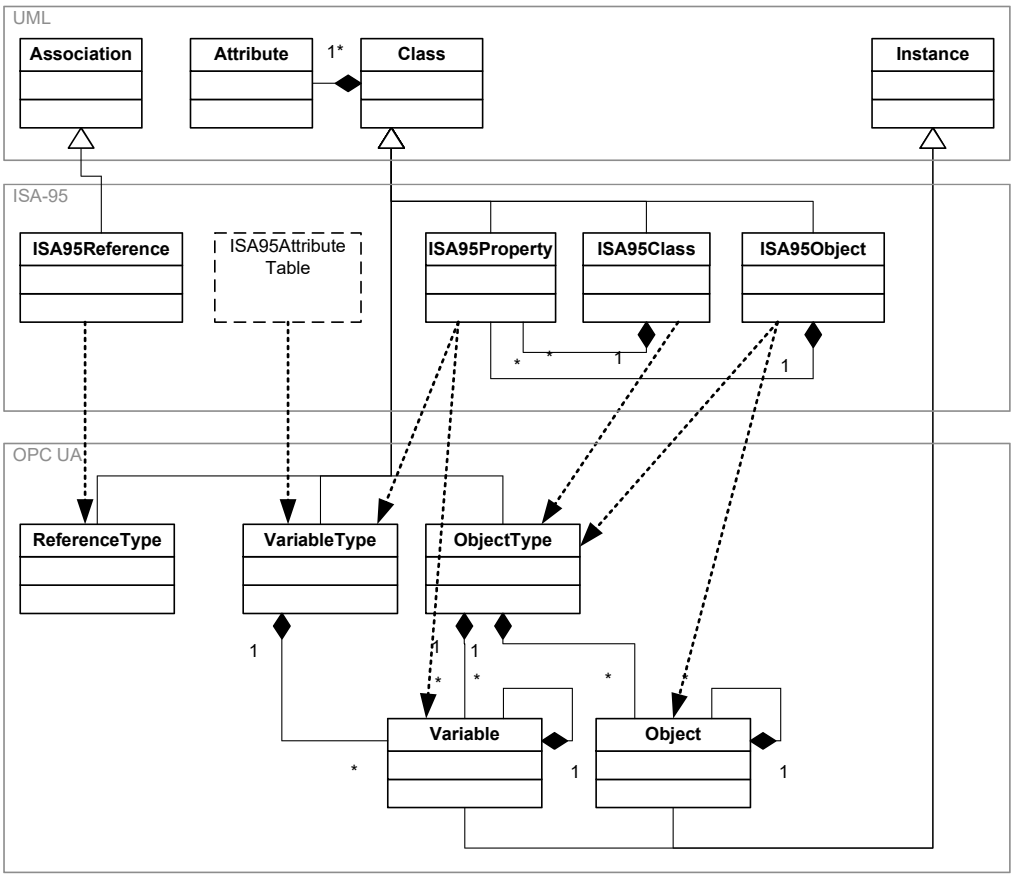
toteutus ISA-95-standardin määrittämästä tietomallista. B2MML on esimerkki toteutuksesta, jonka avulla voidaan siirtää ISA-95-informaatiomallin mukaisesti organisoitua tietoa tuotantolaitoksen eri järjestelmien välillä [3][14, s. 13].

2.3 OPC UA ja ISA-95-liitännäisspesifikaatio

OPC UA -liitännäisspesifikaatioiden avulla voidaan määrittää tietyille toimialoille sopivia OPC UA -informaatiomalleja. Tietyn toimialan informaatiomalli määrittää toimialakohtaiset oliot, muuttujat, tietotyypit sekä viitteet. Muiden liitännäisspesifikaatioiden ohella OPC Foundation -säätiö on määrittänyt ISA-95-liitännäisspesifikaation, jonka avulla voidaan liittää tehtaan tuotannonohjausjärjestelmät korkeamman tason järjestelmiin käyttäen OPC UA -protokollaa sekä ISA-95-tietomallia [5, s. 120–121][14, s. 19].

OPC UA ISA-95-liitännäisspesifikaation tarkoitus on määritellä informaatiomalli, joka on yhteensopiva ISA-95-standardin määrittämän tietomallin sekä OPC UA -protokollan määrittämän tietomallin kanssa [14, s. 1]. Liitännäisspesifikaation tarkoitus onkin tarjota tapa, jonka avulla ISA-95-tietomallinnuksen konseptit voidaan tuoda OPC UA -ympäristöön. OPC UA -standardin laajennos hyödyntää OPC UA:n informaatiomallinnuksen periaatteita ISA-95-standardin oliomallien kuvaamiseen [14, s. 3]. Tämänhetkinen liitännäisspesifikaatio määrittää vain osan ISA-95-standardista. OPC UA ISA-95-liitännäisspesifikaatiossa on kuvattu ISA-95-standardin määrittämät Role Based Equipment, Physical Asset, Material ja Personnel -oliomallit [14, s. 6]. Jotta ISA-95-standardin toteuttavan B2MML -mallin liittäminen OPC UA -ympäristöön olisi mahdollisimman helppoa, liitännäisspesifikaatio pyrkii noudattamaan B2MML-standardin määrittämiä mallinnuskäytäntöjä [14, s. 13].

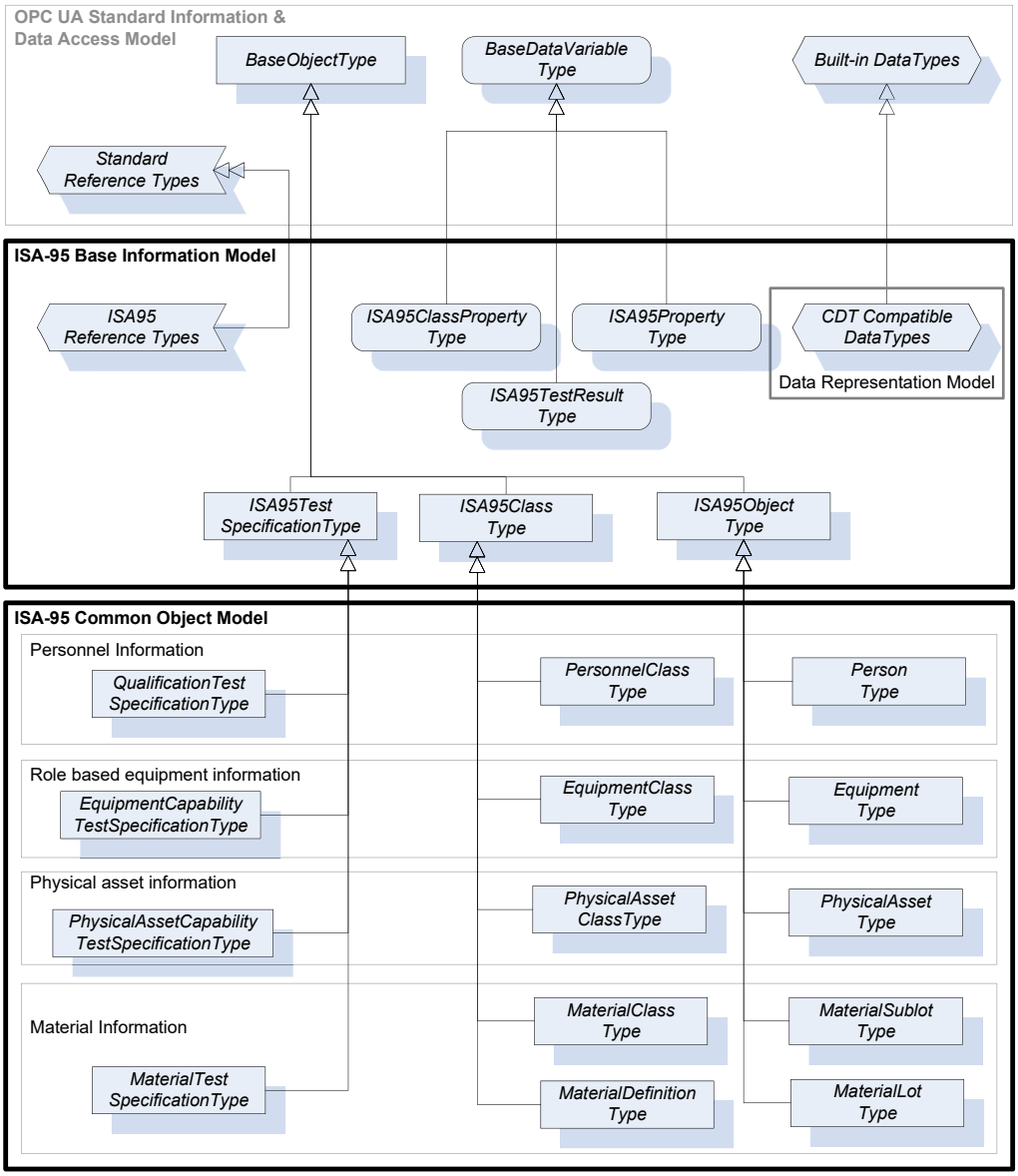
ISA-95-liitännäisspesifikaation keskeinen tarkoitus on määrittää tapa, jolla ISA-95-standardin UML-mallinnuskielellä esitetyt oliomallit voidaan kuvata OPC UA -osoiteavaruuteen [14, s. 22]. ISA-95-standardin määrittämän metamallin ja OPC UA -osoiteavaruusmallin välinen riippuvuus on esitetty kuvassa 2.8.



Kuva 2.8. *ISA-95-metamallin ja OPC UA -osoiteavaruusmallin välinen riippuvuus [14, s. 22].*

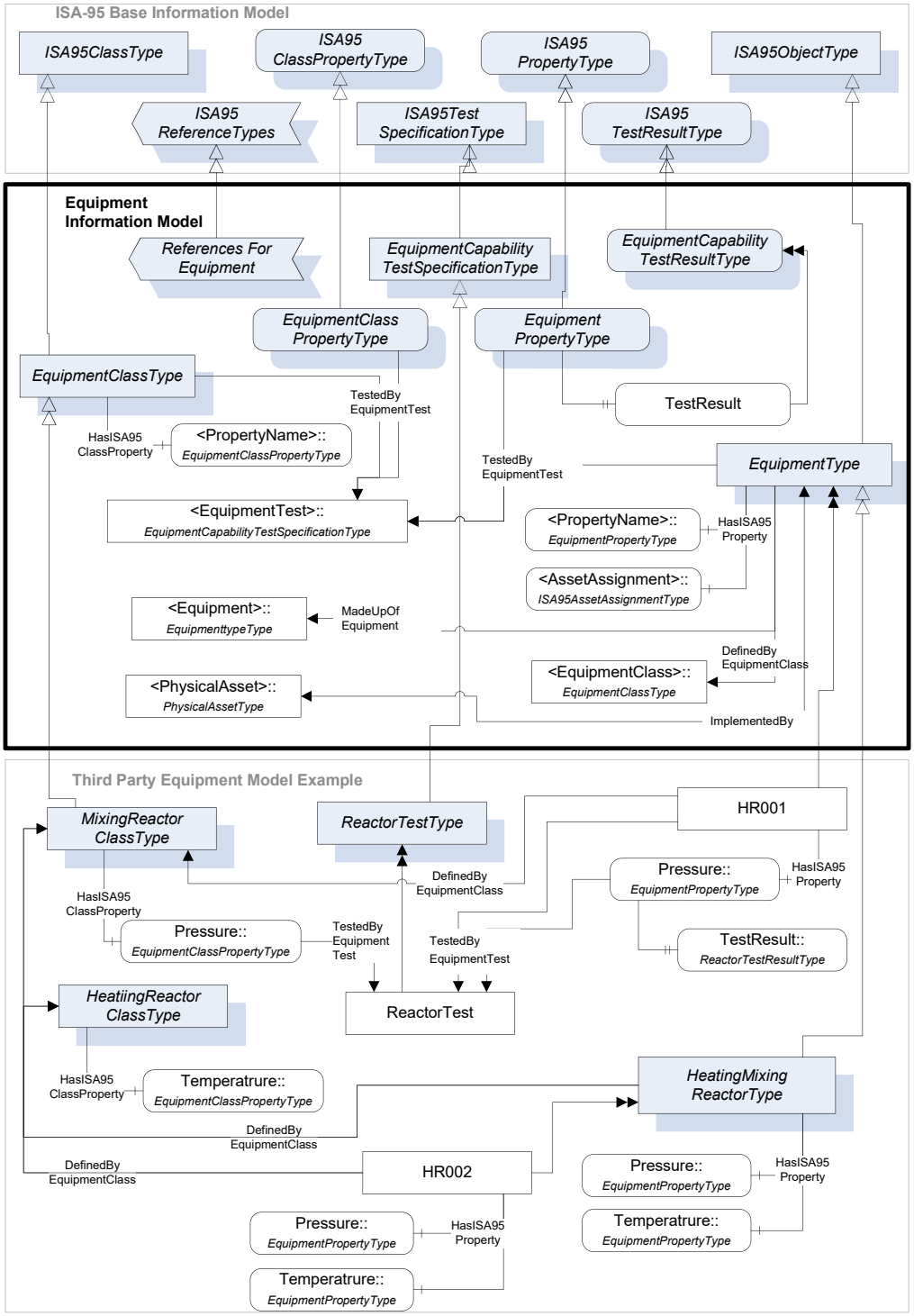
ISA-95-liitännäisspesifikaatio määrittää yleisiä ohjeita, joilla ISA-95-malli voidaan konvertoida OPC UA -malliksi. Keskeisiä kohtia ISA-95-mallin muuttamisessa OPC UA -malliksi ovat esimerkiksi ISA-95-ominaisuuksien (Property) kuvaaminen OPC UA -protokollan muuttujiksi ja muuttujia kuvaaviksi tietotyypeiksi, ISA-95-luokkien kuvaaminen OPC UA ObjectType -tyyppisinä olioina sekä ISA-95-malliin kuuluvien osien välisten riippuvuuksien kuvaaminen OPC UA -viitteiksi. Liitännäisspesifikaatiossa todetaan, että ISA-95-mallin muuttamisessa OPC UA -informaatiomallin mukaiseksi tulee noudattaa olemassa olevia OPC UA:n mallinnustapoja ja käytäntöjä [14, s. 21].

OPC UA ISA-95-liitännäisspesifikaatiossa on määritetty yleinen informaatiomalli (Base information model), jonka tehtävä on kuvata liitännäisspesifikaatiossa käytettävät tietomallit [14, s. 27]. Kuvassa 2.9 on esitetty liitännäisspesifikaation määrittämä ISA-95-informaatiomalli. Kuvasta voidaan nähdä, miten OPC UA -informaatiomalli liittyy ISA-95-informaatiomalliin ja kuinka ISA-95-informaatiomalli määrittää ISA-95-standardiin liittyvät yleiset oliomallit (ISA-95 Common Object Model).



Kuva 2.9. *ISA-95-informaatiomalli [14, s. 27].*

Role based equipment - ja Physical asset -oliomallit on kuvattu liitännäisspesifikaatiossa ja kyseisiä oliomalleja käytetään opinnäytetyön osana tehtävässä OPC UA -palvelimessa. Kuvassa 2.10 on esitettynä liitännäisspesifikaation määrittämä Equipment-informaatiomalli. Keskeisiä osia Equipment-informaatiomallissa ovat EquimentType- sekä EquipmentClass-Type-solmut [14, s. 54]. Physical Asset -informaatiomallissa PhysicalAssetClassType- ja PhysicalAssetType-solmut määrittävät tietotyypit, joista uusia tyyppimäärittelyitä kuvaavia solmuja voidaan periyttää sekä fyysisiä laitteita kuvaavia olioita instantioida [14, s. 62].



Kuva 2.10. *ISA-95 Equipment-informaatiomalli [14, s. 55].*

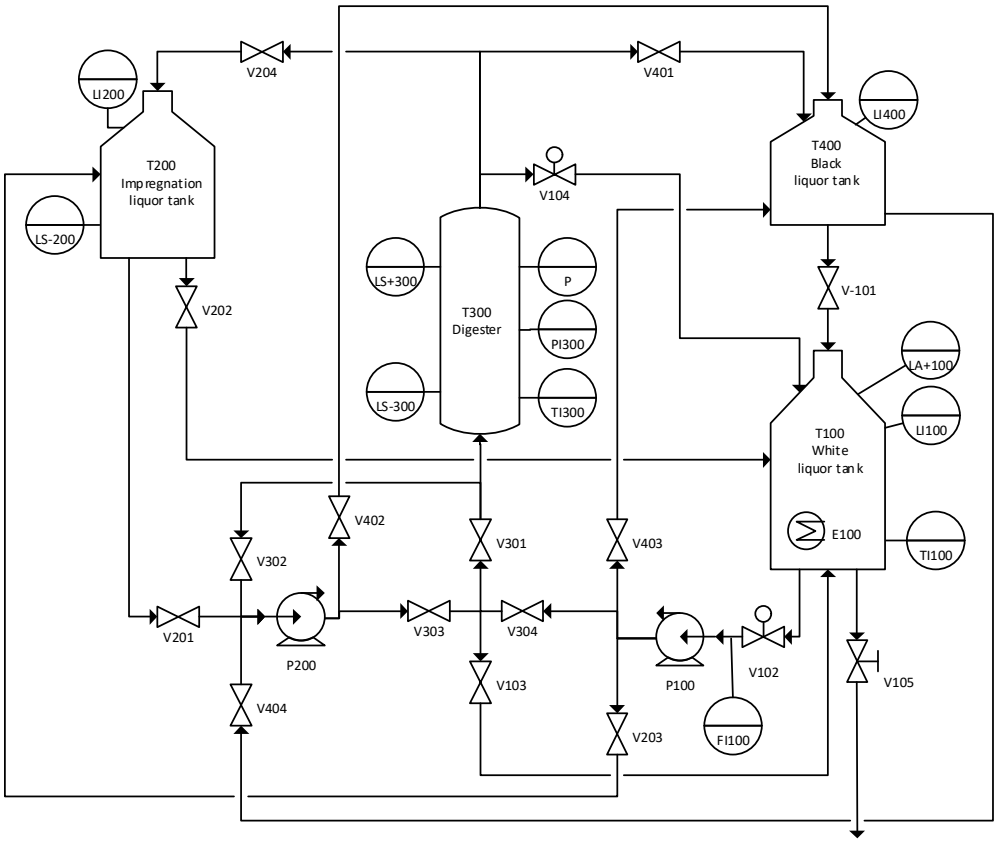
2.4 Panosprosessi

Opinnäytetyössä hyödynnetään sellunkeittoprosessia simuloivaa panosprosessilaitteistoa. Laitteiston tarkoitus on mallintaa sellunkeittoprosessia sen eri vaiheineen. Opinnäytetyön kannalta kiinnostava osa minipanosprosessia on prosessia ohjaava Beckhoff CX

2031 -teollisuus-PC. Kyseiselle teollisuus-PC:lle on määritelty OPC UA -palvelin, jonka avulla panosprosessiin liittyville laitteille voidaan esimerkiksi sekä siirtää ohjaustietoa että lukea laitteiden tallentamaa mittaustietoa.

2.4.1 Prosessilaitteet

Panosprosessilaitteisto koostuu nestetankeista, tankkeja yhdistävästä putkistosta, eri toimilaitteista sekä antureista. Panosprosessilaitteiston rakenne prosessilaitteiden tunnuksineen on esitetty kuvan 2.11 putkisto- ja instrumentointikaaviossa (PI-kaaviossa).



Kuva 2.11. *Panosprosessia kuvaava PI-kaavio [7].*

Sellunkeittoprosessilaitteistossa keskeisessä osassa ovat prosessinesteille tarkoitetut säiliöt sekä keittokattila. Valkolipeää varten järjestelmässä on tankki T100, kyllästeelle on varattu tankki T200 ja mustalipeää varten on tankki T400. Tankki T300 on keittokattila, jonka kautta prosessinesteitä kierrätetään prosessin eri vaiheissa [7].

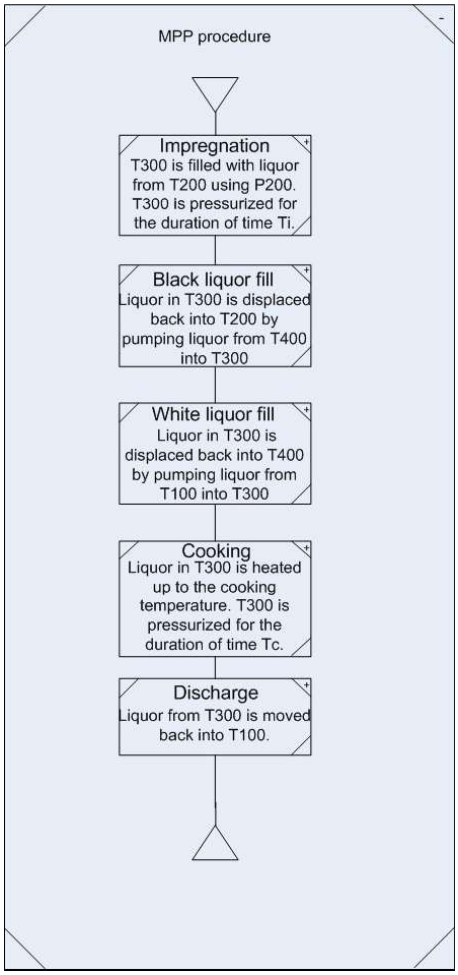
Panosprosessiin liittyviä toimilaitteita ovat venttiilit, pumput sekä lämmitin. Prosessissa on kahden tyyppisiä venttiileitä: sulku- sekä säätöventtiileitä. Sulkuventtiileitä käytetään prosessissa esimerkiksi nestevirtauksien ohjaamiseen sekä kattiloiden paineiden hallintaan. Säätöventtiileitä käyttäen prosessissa toteutetaan T300-tankin paineen säätö. Prosessi-

laitteiston pumpuilla venttiileitä apuna käyttäen kierrätetään prosessin nesteitä eri tankkien välillä. Prosessilaitteiston E100-lämmitintä käytetään valkolipeän lämpötilan nostamiseen [7].

Prosessilaitteistoon liittyviä antureita ovat virtausanturi FI100, paineanturi PI300, lämpötila-anturit TI100 ja TI300 sekä pinnankorkeusanturit LI100, LI200 ja LI400. Lisäksi laitteistossa on pintahälytin LA100, joka hälyttää, mikäli T100-tankin pinnankorkeus nousee sallittua rajaa ylemmäksi. Lisäksi tankeissa T200 ja T300 on pintavahtityyppiset anturit LS-200, LS-300 sekä LS+300, joiden ulostulo on 1, mikäli prosessinesteen pinta ylittää anturin mittauspisteen, tai 0, mikäli prosessinesteen pinta alittaa anturin mittauspisteen [7].

2.4.2 Sellunkeittoprosessi

Sellunkeittoprosessi koostuu viidestä eri vaiheesta, jotka ovat Impregnation, Black liquor fill, White liquor fill, Cooking sekä Discharge. Kuvassa 2.12 on esitettynä PFC-kaavio, joka kuvaa sellunkeittoprosessiin liittyviä vaiheita [7].



Kuva 2.12. *Panosprosessin eri vaiheita kuvaava PFC-kaavio [7].*

Kyllästysvaiheessa prosessilaitteiston neste pumpataan keittokattilaan, jonka painetta nostetaan ennalta määritetyn aikajakson aikana. Kun painetta on kasvatettu riittävän kauan, siirrytään Black liquor fill -vaiheeseen. Black liquor fill -vaiheessa keittokattilan painetta lasketaan ja keittokattilassa oleva kylläste korvataan mustalipeällä (Black liquor). Mustalipeää pumpataan keittokattilaan mustalipeäsäiliöstä niin kauan, että keittokattilassa olevan mustalipeän pinnankorkeus saavuttaa ennalta määrätyn tavoitearvonsa. Kun ennalta määritetty pinnankorkeus saavutetaan, siirrytään sellunkeittosekvenssissä White liquor fill -osaan. Tässä sekvenssin vaiheessa keittokattilassa oleva mustalipeä korvataan valkolipeätankista pumpattavalla valkolipeällä (White liquor). Kuten Black liquor fill -sekvenssin tapauksessa, valkolipeää pumpataan keittokattilaan niin kauan, että ennalta määritetty pinnankorkeus saavutetaan. Seuraavassa sekvenssin osassa eli Cooking-vaiheessa valkolipeää aletaan kierrättämään valkolipeätankin sekä keittokattilan kautta. Lipeän kierrätyksen yhteydessä valkolipeän lämpötilaa nostetaan ennalta määrättyyn tavoitelämpötilaan. Kun tavoitelämpötila saavutetaan, ylläpidetään lämpötilaa ennalta määrätyn keittoajan verran. Kun lipeää on keitetty riittävän kauan, suoritetaan sellunkeittosekvenssin Discharge-vaihe, jossa valkolipeä pumpataan takaisin valkolipeätankkiin. Discharge-vaiheen jälkeen sellunkeittosimulaattori on alkutilassaan [7].

2.4.3 OPC UA -palvelimen tietomalli

Panosprosessia ohjaavalle Beckhoff CX 2031 -teollisuus-PC:lle on prosessin ohjauslogiikan lisäksi konfiguroitu OPC UA -palvelin, joka mahdollistaa sekä ohjaustiedon välittämisen prosessille että mittaustietojen lukemisen prosessilta. Sellunkeittoprosessista ja sitä ohjaavasta OPC UA -palvelimesta on toteutettu ohjelmistosimulaattori, jonka OPC UA -palvelimen tietomalli on todellista prosessia ohjaavan OPC UA -palvelimen tietomallin alijoukko. Vaikka simulaattorin OPC UA -palvelimen osoiteavaruus ei vastaa täysin fyysistä prosessia ohjaavan OPC UA -palvelimen osoiteavaruutta, voidaan prosessi toteuttaa käyttäen simulaattorisovellusta. Opinnäytetyössä hyödynnetään fyysistä prosessia mallintavaa simulaattoria. Sellunkeittosimulaattorin sisältämän OPC UA -palvelimen tietomalli on esitetty kuvassa 2.13.

Kuva 2.13. *UaExpert-asiakassovelluksen näkymä sellunkeittoprosessia mallintavan simulaattorin OPC UA -osoiteavaruuteen, jossa on esitetty panosprosessia ohjaavan OPC UA -palvelimen tietomallin rakenne.*

Panosprosessisimulaattorin OPC UA -palvelimen osoiteavaruuden perusta on Root-olio, jonka tehtävä on toimia pisteenä, jonka kautta palvelimen sisältävää osoiteavaruutta päästään tarkastelemaan. Root-olio pitää sisällään Objects-, Types- ja Views-oliot. Objects-olion tarkoitus on tarjota alkupiste, jonka kautta olioita ja muuttujia, jotka eivät tarjoa tyyppimäärittelyä, päästään tarkastelemaan. Types-olio pitää sisällään tyyppimäärittelyihin käytettävät oliot. Views-olion kautta päästään tarkastelemaan OPC UA -palvelimeen toteutettuja näkymiä. Edellä mainitut FolderType-tyyppiset oliot ovat OPC UA -standardissa määritetty sellaisiksi solmuiksi, jotka tulee olla jokaisessa OPC UA -palvelintoteutuksessa [11, s. 45–47].

Sellunkeittoprosessia mallintavan simulaattorisovelluksen OPC UA -palvelimeen ei ole toteutettu näkymiä. Palvelin hyödyntää lisäksi OPC UA -standardin määrittämiä perustietotyyppejä. Objects-olio sisältää ServerType-tyyppisen Server-olion sekä BaseObject-

Type-tyyppisen PLC1-olion. Server-olio pitää sisällään palvelimen parametreja ja muuta informaatiota. Server-olion kautta päästään tarkastelemaan palvelimen tietoja [11, s. 53]. PLC1-olio sisältää FolderType-tyyppisen eq_states-olion, jolle on määritetty muuttujia, jotka pitävät sisällään tietoa panosprosessin eri toimilaitteiden tilasta sekä antureiden mittausarvoista.

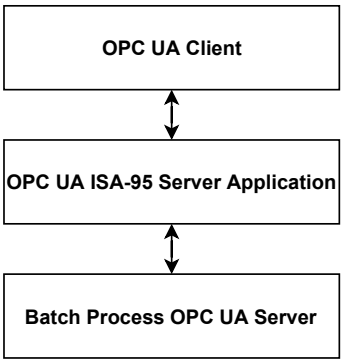
3 OPC UA -PALVELIMEN TOTEUTUS

3.1 Ohjelman rakenne

Opinnäytetyön osana toteutetaan ISA-95-liitännäisspesifikaation mukainen palvelin, jonka avulla sellunkeittosimulaattoria ohjaavan OPC UA -palvelimen tietomalli voidaan muuttaa tukemaan ISA-95-standardin määrittämää tiedonkuvaustapaa. OPC UA ISA-95-palvelinohjelma toimiikin sovelluksena, jonka avulla sellunkeittosimulaattoriin liittyvällä OPC UA -palvelimella olevien muuttujien arvoja voidaan kirjoittaa ja lukea.

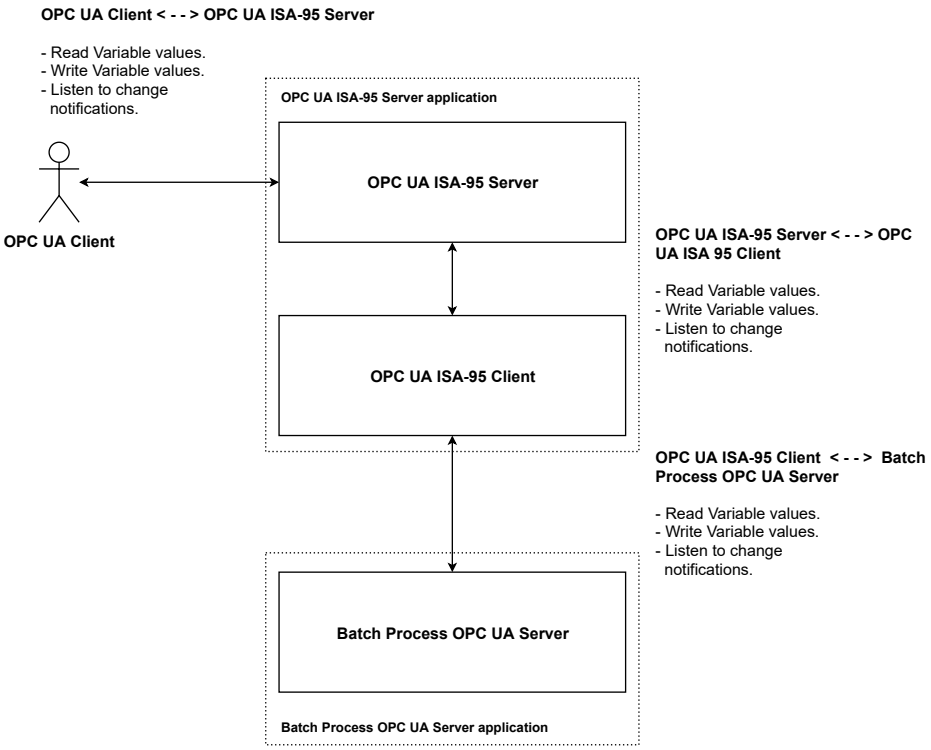
Kuvassa 3.1 on esitetty toteutettava palvelinsovellus ja siihen keskeisesti liittyvät muut sovellukset. Toteutettavalla OPC UA ISA-95-palvelinsovelluksella (kuvan 3.1 OPC UA ISA-95 Server Application) on kaksi päätehtävää, jotka ovat ISA-95-liitännäisspesifikaation mukaiseen OPC UA -palvelimeen liittyvien toiminnallisuuksien tarjoaminen sekä tiedon siirtäminen OPC UA ISA-95-palvelimen ja sellunkeittosimulaattoria ohjaavan OPC UA -palvelimen (kuvan 3.1 Batch Process OPC UA Server) välillä. Toteutettava sovellus yhdistää OPC UA ISA-95-palvelimen osoiteavaruudessa olevat muuttujat niitä vastaaviin sellunkeittosimulaattoria ohjaavan palvelimen osoiteavaruudessa oleviin muuttujiin. Näin ollen OPC UA ISA-95-palvelinsovellus synkronoikin oman tilansa sellunkeittosimulaattoria ohjaavan OPC UA -palvelimien kanssa. Jotta OPC UA ISA-95-palvelinsovellus pystyy ylläpitämään palvelimien tilaa, tulee sovelluksen reagoida OPC UA ISA-95-palvelimen osoiteavaruuden muuttujissa sekä panosprosessia mallintavan ohjelmistosimulaattorin OPC UA -palvelimen muuttujissa tapahtuviin muutoksiin.

Kuvassa 3.1 esitetty OPC UA Client on OPC UA -asiakassovellus, jolla OPC UA ISA-95-palvelinsovelluksen tarjoamaa OPC UA -osoiteavaruutta voidaan tarkastella ja muokata. Asiakassovelluksella voidaan esimerkiksi muuttaa OPC UA ISA-95-palvelimen osoiteavaruuden muuttujien arvoja. Asiakassovellus, jonka avulla voidaan tarkastella ja muokata OPC UA ISA-95-palvelinsovelluksen OPC UA -osoiteavaruutta, ei ole osa opinnäytetyössä toteutettavaa sovellusta.



Kuva 3.1. *Opinnäytetyössä toteutettava OPC UA ISA-95 Server -sovellus sekä siihen keskeisesti liittyvät sovellukset OPC UA Client ja Batch Process OPC UA Server.*

Kuvassa 3.2 on esitetty tarkemmin palvelinsovelluksen rakenne sekä systeemin eri osien välillä liikkuva informaatio. Palvelinsovellus koostuu kahdesta pääkomponentista, jotka ovat OPC UA ISA-95 Server ja OPC UA ISA-95 Client. OPC UA ISA-95 Server toteuttaa OPC UA ISA-95-liitännäisspesifikaation mukaisen palvelimen. OPC UA ISA-95 Client -osan tehtävä on toimia tiedonvälittäjäkomponenttina systeemiin liittyvien OPC UA -palvelimien välillä. Sovellukseen liittyvän OPC UA ISA-95-asiakassovelluskomponentin tehtävä on rekisteröityä kuuntelemaan kummankin OPC UA -palvelimen muuttujissa tapahtuvia muutoksia. Muutosten yhteydessä asiakassovelluskomponentti välittää muutoksen tiedot toiselle palvelimelle, jotta palvelimien vastaavien muuttujien arvot pysyisivät yhdenmukaisina. Kuvassa 3.2 on lisäksi esitetty systeemiin kuuluvien asiakassovellusten toiminnallisuus sekä niiden hyödyntämä tieto. Asiakassovellusten keskeiset toiminnallisuudet ovat palvelimilla olevien muuttujien arvojen lukeminen ja kirjoittaminen. Asiakassovellukset kuuntelevat lisäksi muuttujien arvojen vaihtuessa syntyviä muutosilmoituksia. Systeemin eri komponenttien välillä liikkuukin näin ollen pääasiassa tietoa palvelimiin liittyvistä muuttujista.



Kuva 3.2. *Palvelinsovelluksen rakenne ja sovelluksen eri komponenttien väliset informaatiovirrat.*

3.2 Palvelimen toteutus

Opinnäytetyössä toteutettu ISA-95-liitännäisspesifikaation mukainen OPC UA -palvelin on implementoitu käyttäen Node.js -ympäristöä sekä node-opcua ja node-opcua-isa95 -moduuleita. Node.js on tarkoitettu verkottuneiden sovellusten toteuttamiseen JavaScript-ohjelmointikielellä [8], kun taas palvelimen toteutuksessa käytettävät node-opcua ja node-opcua-isa95 ovat Node.js-ympäristöön tarkoitettuja ohjelmistopaketteja, jotka tarjoavat työkalut, joiden avulla voidaan toteuttaa OPC UA -sovelluksia. Node-opcua -paketti tarjoaa toiminnallisuuden, jonka avulla voidaan toteuttaa OPC UA -palvelimia sekä -asiakassovelluksia. Node-opcua-isa95 -ohjelmistopaketti tarjoaa toiminnallisuuden, jonka avulla voidaan toteuttaa ISA-95-liitännäisspesifikaation mukaisia OPC UA -sovelluksia [12][13]. OPC UA ISA-95-palvelimen toteutuksessa käytetään Node.js ympäristön versiota 10.16.0, node-opcua -paketin versiota 0.7.1 sekä node-opcua-isa95 -paketin versiota 0.2.0.

Palvelinsovelluksen toteuttamiseksi tulee asentaa Node.js -ympäristö. Node.js ympäristön asennuksen jälkeen asennetaan OPC UA -sovellusten kehittämiseen tarvittavat node-opcua ja node-opcua-isa95 -paketit.

Ohjelma 3.1 kuvaa palvelimen alustamisen keskeisimmät kohdat. Riveillä 1–4 valitaan käytettävät ohjelmistomoduulit. Tarvittavat ohjelmistokomponentit ovat edellä mainitut OPC UA -paketit sekä OpcUaIsa95Client-luokka, joka tarjoaa toiminnallisuuden, jonka avulla järjestelmään liittyviin OPC UA -palvelimiin voidaan liittyä. Käytettävien ohjelmisto-

moduulien valitsemisen jälkeen määritetään osoiteavaruudessa käytettävät solmut. Viitteet käytettäviin solmuihin lisätään rivillä 7 xmlFiles-muuttujaan tallennettuun taulukkoon. Solmut on määritetty node-opcua ja node-opcua-isa95 -moduuleissa XML-tiedostoina [12][13]. Rivillä 13 luodaan uusi OPCUAServer-instanssi. Palvelininstanssille annetaan JavaScript object -tyyppisessä rakenteessa tarvittavat parametrit, kuten palvelimen portti, osoiteavaruudessa käytettävät solmut, sekä polku, jonka avulla palvelimen sisältämään tietoon päästään käsiksi. Viimeisellä lähdekoodirivillä OPCUAServer-instanssi alustetaan. Alustusfunktiolle annetaan parametrina post_initialize-funktio, joka suoritetaan, kun palvelimen alustus on valmis. Post_initialize-funktiossa suoritetaan osoiteavaruuden luominen sekä OPC UA ISA-95-palvelinsovellukseen liittyvien asiakassovellusten luominen. OPC UA -asiakassovellusinstanssien käyttäminen tapahtuu riveillä 38–44. Ensimmäinen vaihe OpcUaIsa95Client-luokan käyttöä on luoda yhteys palvelinsovelluksiin. Yhteyden muodostaminen tapahtuu metodilla connectClients. Yhteyden muodostamisen jälkeen OpcUaIsa95Client-luokan sisältämille OPCUAClient-luokan instansseille luodaan istunnot. Viimeiseksi OPCUAClient-instanssit liitetään kuuntelemaan palvelimien osoiteavaruuksien olioiden muutoksia sekä määritetään ne solmut, joiden muutosilmoituksia halutaan kuunnella. Muutosilmoituksia liitytään kuuntelemaan OpcUaIsa95Client-luokan metodilla addSubscriptions, ja ne solmut, joita halutaan monitoroida, määritetään metodissa addMonitoredItems.

```
1  // Lisaa tarvittavat ohjelmistomodulit.
2  const OpcUaIsa95Client = require("./OpcUaIsa95Client");
3  const opcua = require("node-opcua");
4  require("node-opcua-isa95")(opcua);
5
6  // Osoiteavaruudessa kaytettavien solmujen maarittelyt.
7  const xmlFiles = [
8      opcua.standard_nodeset_file,
9      opcua.ISA95.nodeset_file
10 ];
11
12 // Luo OPCUAServer instanssi.
13 const server = new opcua.OPCUAServer({
14     port: 4334,
15     nodeset_filename: xmlFiles,
16     resourcePath: "UA/OpcUaISA95BatchProcessServer",
17     buildInfo : {
18         productName: "OPC UA ISA-95 Batch Process Server"
19     }
20 });
21
22 // Kun palvelin on alustettu, kutsutaan
23 // post_initialize-funktiota.
```

```
24 function post_initialize() {
25     // Luo palvelimen osoiteavaruus.
26     const addressSpace = server.engine.addressSpace;
27     generateAddressSpece(addressSpace);
28
29     // Kaynnista OPC UA -palvelin
30     server.start(async function() {
31         const endpointUrl = server.endpoints[0]
32                                 .endpointDescriptions()[0]
33                                 .endpointUrl;
34         // Luo OPC UA asiakassovellusinstanssit,
35         // jotka muodostavat yhteyden OPC UA
36         // ISA-95 -palvelimeen seka Batch process
37         // OPC UA -palvelimeen.
38         const opcUaIsa95Client =
39             new OpcUaIsa95Client(opcua);
40         opcUaIsa95Client.generateClients()
41         await opcUaIsa95Client.connectClients();
42         await opcUaIsa95Client.createSessions();
43         await opcUaIsa95Client.addSubscriptions();
44         await opcUaIsa95Client.addMonitoredItems();
45     });
46 }
47
48 // Alusta OPC UA -palvelin
49 server.initialize(post_initialize);
```

Ohjelma 3.1. *Palvelinsovelluksen alustamisen keskeisimmät kohdat.*

GenerateAddressSpace-funktiossa luodaan palvelimen osoiteavaruuden sisältö. Kyseisessä funktiossa luodaan osoiteavaruuden tyyppimäärittelyt sekä instantioidaan sellunkeittosimulaattoriprosessin loogisia ja fyysisiä laitteita kuvaavat oliot.

Ohjelma 3.2 kuvaa, kuinka osoiteavaruuden Equipment-hierarkiaan liittyvät Equipment Level -arvot määritetään. Osoiteavaruuteen luodaan uusi EquipmentClassType, jolle määritetään luonnin yhteydessä equipmentLevel-ominaisuudeksi haluttu Equipment Level -arvo. Tämän seurauksena luodulle EquipmentClassType-tyyppiselle solmulle syntyy EquipmentLevel-attribuutti. Esimerkissä osoiteavaruuteen on luotu uusi EquipmentClassType, jolle on määritetty Equipment Level -arvoksi Process Cell. Luotua EquipmentClassType-tyyppimäärittelyä voidaan käyttää määrittämään Equipment-olioinstanssia. Ohjelmassa 3.2 luodaan uusi Equipment-olioinstanssi, jolle annetaan luonnin yhteydessä definedByEquipmentClass-ominaisuus. Tämän ominaisuuden arvoksi asetetaan haluttu Equip-

mentClassType-määrittely. Uuteen Equipment-instanssiin on luonnin yhteydessä määritetty myös containedByEquipment-ominaisuus, jolla kuvataan Equipment-olioinstanssin kuulumista toiseen Equipment-instanssiin.

```
1  const processCellEquipmentClassType =
2  addressSpace.addEquipmentClassType({
3      browseName: "ProcessCellEquipmentClassType",
4      equipmentLevel: EquipmentLevel.ProcessCell
5  });
6
7  const simulatorCellEquipment = addressSpace.addEquipment({
8      browseName: "SimulatorCell",
9      containedByEquipment: laboratoryEquipment,
10     definedByEquipmentClass: processCellEquipmentClassType
11 });
```

Ohjelma 3.2. *Equipment Level -arvon määrittäminen.*

Ohjelmissa 3.3–3.6 on kuvattu, miten sellunkeittosimulaattoriin liittyvä pinnankorkeusanturi voidaan mallintaa OPC UA ISA-95-palvelimen osoiteavaruuteen. Sellunkeittosimulaattorin muiden laitteiden mallintaminen OPC UA ISA-95-palvelimen osoiteavaruuteen voidaan toteuttaa vastaavasti kuin ohjelmissa 3.3–3.6 on esitetty.

Ohjelmassa 3.3 on esitetty, kuinka pinnankorkeusanturia kuvaava EquipmentClassType-tyyppimäärittely kuvataan osoiteavaruuteen. Uudelle EquipmentClassType-tyypille määritetään pelkästään browseName-parametri. BrowseName-ominaisuudella määritetään olion nimi, jolla kyseiseen olioon voidaan viitata palvelimen osoiteavaruudessa. Ohjelmassa on esitetty myös uuden Equipment-olioinstanssin luominen. Osoiteavaruuteen voidaan lisätä uusia Equipment-instansseja addEquipment-funktiolla. Ohjelmassa 3.3 uudelle instanssille on annettu browseName-parametrin arvoksi LI100. DefinedByEquipmentClass-ominaisuutta käytetään kuvaamaan se EquipmentClassType-tyyppinen olio, joka määrittää Equipment-instanssia. Ohjelmassa 3.3 on uuden equipment-olioinstanssin luonnissa käytetty myös parametria containedByEquipment, jolla määritetään se Equipment-instanssi, johon uusi olioinstanssi kuuluu.

```
1  const levelIndicatorEquipmentClassType =
2  addressSpace.addEquipmentClassType({
3      browseName: "LevelIndicatorEquipmentClassType"
4  });
5
6  const li100Equipment = addressSpace.addEquipment({
7      browseName: "LI100",
8      containedByEquipment: t100Equipment,
9      definedByEquipmentClass: levelIndicatorEquipmentClassType
10 });
```

Ohjelma 3.3. *EquipmentClassType-määrittelyn toteuttaminen ja Equipment-instanssin instantiointi palvelimen osoiteavaruuteen.*

Ohjelmassa 3.4 kuvataan sekä pinnankorkeusanturiin liittyvän PhysicalAssetClassType-tyyppisen olion että PhysicalAssetClassType-olioon liittyvän ClassProperty-ominaisuuden määrittäminen. Sellunkeittosimulaattoriin liittyvien pinnankorkeusantureiden kuvaamiseen käytettävän PhysicalAssetClassType-olion browseName-ominaisuudeksi asetetaan Level Indicator. ModelNumber-ominaisuutta käytetään määrittämään pinnankorkeusanturin mallinumero. Manufacturer-ominaisuudella kuvataan laiteen valmistaja. Valmistaja määritetään JavaScript-oliolla, jolla on ominaisuudet dataType sekä value. DataType-ominaisuudella määritetään osoiteavaruuteen syntyvän valmistajaa kuvaavan Variable-solmun sisältämän tiedon tietotyyppi. Value-ominaisuudella määritetään muuttujan sisältämän merkkijonon sisältö.

Pinnankorkeusantureita määrittävälle PhysicalAssetClassType-oliolle on määritetty ISA95-ClassPropertyType-tyyppinen solmu, jota käytetään kuvaamaan pinnankorkeusanturin mittaamaa lukemaa. ISA95ClassPropertyOf-ominaisuudella määritetään se PhysicalAssetClassType-olio, johon ominaisuus kuuluu, browseName-ominaisuudella määritetään solmun nimi osoiteavaruudessa ja dataType-ominaisuudella kuvataan luotavan ominaisuussolmun sisältävän tiedon tietotyyppi. ISA95ClassPropertyType-tyyppisen solmun luonnissa määritetään lisäksi value-ominaisuudella luotavan solmun sisältämä tieto sekä modellingRule-ominaisuudella se, onko luotava parametri pakollinen vai valinnainen osa instantioitavaa oliota.

```
const levelIndicatorPhysicalAssetClassType =
addressSpace.addPhysicalAssetClassType({
    browseName: "Level indicator",
    modelNumber: "Level indicator 1",
    manufacturer: {
        dataType: "String",
        value: {
            dataType: opcua.DataType.String,
            value: "Tuni – Tampere universities"
        }
    }
});

addressSpace.addISA95ClassProperty({
    ISA95ClassPropertyOf: levelIndicatorPhysicalAssetClassType,
    typeDefinition: "ISA95ClassPropertyType",
    browseName: "Level",
    description: "Liquor level, 0..35 mm",
    dataType: "UInt16",
    value: {dataType: opcua.DataType.UInt16, value: 0},
    modellingRule: "Mandatory"
});
```

Ohjelma 3.4. *PhysicalAssetClassType-tyypin toteuttaminen palvelimen osoiteavaruuteen.*

Sellunkeittosimulaattorin pinnankorkeusantureille on määritetty oma PhysicalAssetType-tyyppinen olio sekä PhysicalAssetType-olioon liittyvä ISA95PropertyType-tyyppinen ominaisuus, jolla kuvataan fyysisen pinnankorkeusanturin mittausarvoa. PhysicalAssetType-tyyppisen olion sekä PhysicalAssetType-olioon liittyvän ISA95PropertyType-tyyppisen ominaisuuden määrittäminen on esitetty ohjelmassa 3.5.

```
const levelIndicatorPhysicalAssetType =
addressSpace.addPhysicalAssetType({
    browseName: "levelIndicatorPhysicalAssetType"
});

addressSpace.addISA95Property({
    browseName: "Level",
    dataType: "UInt16",
    value: { dataType: opcua.DataType.UInt16, value: 0 },
    ISA95PropertyOf: levelIndicatorPhysicalAssetType,
    modellingRule: "Mandatory",
    typeDefinition: "ISA95PropertyType",
});
```

Ohjelma 3.5. *PhysicalAssetType-tyypin toteuttaminen palvelimen osoiteavaruuteen.*

Ohjelmassa 3.6 on kuvattu fyysistä laitetta kuvaavan PhysicalAsset-instanssin luominen. Osoiteavaruuteen uusi järjestelmän fyysistä laitetta kuvaava olioinstanssi luodaan node-opcus-isa95 -ohjelmistomoduulin addPhysicalAsset-funktiolla. Luotava instanssi määritellään JavaScript-oliolla, jolla on seuraavat ominaisuudet: containedByPhysicalAsset, definedByPhysicalAssetClass, typeDefinition, browseName, description, modellingRule, vendorId, fixedAssetId ja implementationOf. ContainedByPhysicalAsset-ominaisuudella määritetään se fyysinen laite, jonka osa luotava PhysicalAsset-instanssi on. DefinedByPhysicalAssetClass-ominaisuus määrittää sen PhysicalAssetClassType-määrittelyn, jota käytetään määrittämään luotavaa PhysicalAsset-instanssia. DefinedByPhysicalAssetClass-ominaisuuden arvona käytetään ohjelmassa 3.4 kuvattua PhysicalAssetClassType-määrittelyä. TypeDefinition-ominaisuutta käytetään määrittämään se PhysicalAssetType-tyyppimäärittely, joka kuvaa luotavaa PhysicalAsset-instanssia. Ohjelmassa 3.6 TypeDefinition-ominaisuuden arvona käytetään ohjelmassa 3.5 luotua physicalAssetType-tyyppimäärittelyä. BrowseName-ominaisuus määrittää olion nimen osoiteavaruudessa. Description-ominaisuutta käytetään määrittämään instanssiin liittyvä kuvaus. ModellingRule-ominaisuutta käytetään määrittämään tieto instanssin pakollisuudesta osoiteavaruutta luotaessa. VendorId-ominaisuudella identifioidaan fyysiseen laitteeseen liittyvä toimittaja [14, s. 68]. FixedAssetId-ominaisuudella määritetään fyysistä laitetta identifioiva tunnus. Laitteen toimittaja määrittää FixedAssetId-identifiointinumeron [14, s. 68]. Opinnäytetyössä toteutetussa OPC UA ISA-95-palvelimessa fixedAssetId-ominaisuus määritetään generateAssetId-funktiossa, jossa jokainen osoiteavaruuteen luotava PhysicalAsset-instanssi saa arvokseen jonkin nollaa suuremman kokonaisluvun. ImplementationOf-ominaisuuden avulla määritetään viite loogisen laitteen ja fyysisen laitteen välille. Ohjelmassa 3.6 implementationOf-ominaisuuden arvona käytetään ohjelmassa 3.3 luotua Equipment-instanssia. ImplementationOf-ominaisuuden asettamisen seurauksena Equipment-instanssin ja PhysicalAsset-instanssin välille syntyy ImplementationOf ja ImplementedBy-viitteet.

```
const li100PhysicalAsset = addressSpace.addPhysicalAsset({
    containedByPhysicalAsset: whiteLiquorTankPhysicalAsset,

    definedByPhysicalAssetClass:
        levelIndicatorPhysicalAssetClassType,

    typeDefinition: levelIndicatorPhysicalAssetType,
    browseName: "Level indicator - LI100",
    description: "White liquor tank level indicator",
    modellingRule: "Mandatory",
    vendorId: {
        dataType: "String",
        value: {
                dataType: opcua.DataType.String,
                value: "Vendor 1"
        }
    },
    fixedAssetId: generateAssetId(),
    implementationOf: li100Equipment
});
```

Ohjelma 3.6. *PhysicalAsset-instanssin luominen palvelimen osoiteavaruuteen.*

Palvelinsovellukseen on toteutettu taulukko, johon on määritetty OPC UA ISA-95 Server ja Batch Process OPC UA Server -sovellusten osoiteavaruuksien toisiinsa liittyvät solmut. Ohjelmassa 3.7 on esitetty tapa, jolla eri palvelimien sisältämät solmut on liitetty toisiinsa. NodeIdMapping-taulukko koostuu JavaScript-olioista, joilla on kaksi ominaisuutta: isa95ServerNode ja batchProcessServerNode. Kumpikin ominaisuus pitää sisällään JavaScript-olion, jolla määritetään osoiteavaruuden yksittäisen solmun tiedot. IdentifierType-ominaisuus kuvaa osoiteavaruuden solmun määrittävän nodeId-arvon tietotyypin. NodeId-ominaisuus sisältää solmun nodeId-arvon. NamespaceIndex-ominaisuudella määritetään se OPC UA -palvelimen nimiavaruus, jossa kyseinen solmu sijaitsee. NodeIdMapping-taulukon sisältämän tiedon perusteella palvelinsovelluksen sisältämä OPC UA -asiakassovellus osaa päivittää palvelimen osoiteavaruuden oikean solmun tiedot, kun toisessa palvelimessa vastinsolmun arvo muuttuu.

```
1  const NodeIdMapping = [
2      {
3          isa95ServerNode: {
4              identifierType: opcua.NodeIdType.NUMERIC,
5              nodeId: 1159,
6              namespaceIndex: 1
7          },
8          batchProcessServerNode: {
9              identifierType: opcua.NodeIdType.STRING,
10             nodeId: 'eq_states.EQ_LI100',
11             namespaceIndex: 2
12         }
13     }
14 ]
```

Ohjelma 3.7. *Palvelimien vastinsolmujen määrittäminen.*

Kun OPC UA ISA-95-palvelinsovelluksen nimiavaruus ja asiakassovellusosuus on määritetty, voidaan palvelinsovellus käynnistää. Palvelinsovellus käynnistetään Node.js dokumentaation kuvaamalla tavalla.

3.3 Palvelimen tietomalli

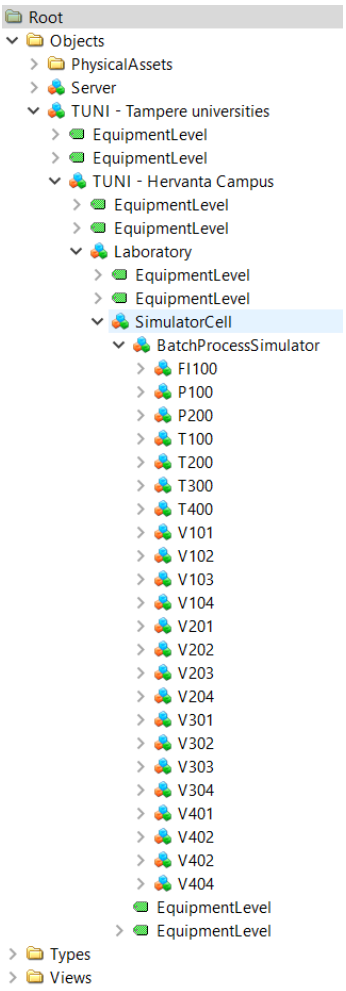
OPC UA ISA-95-palvelimen tietomallin kannalta merkittävimmät osoiteavaruuden osat ovat FolderType-tyyppinen physicalAssets-olio, TUNI-Tampere universities Equipment-olioinstanssi ja TUNI-Tampere universities -olion sisältämät muut Equipment-olioinstanssit. Tietomallin TUNI-Tampere universities -oliota ja sen sisältämiä muita Equipment-olioinstansseja käytetään kuvaamaan ISA-95-standardin määrittämää Role Based Equipment -hierarkiaa. Kaikki järjestelmään liittyvät fyysisiä laitteita kuvaavat olioinstanssit on puolestaan järjestetty PhysicalAssets-olion sisään.

Toteutetun OPC UA ISA-95-palvelimen tietomallin Role Based Equipment -hierarkian, joka on esitetty kuvassa 3.3, ylimmällä tasolla oleva TUNI-Tampere universities -oliota käytetään kuvaamaan tuotanto-organisaation ylintä tasoa. Osoiteavaruuteen onkin määritetty EquipmentClassType-tyyppinen EnterpriseEquipmentClassType, jota käytetään määrittämään Equipment-olioinstansseja, joiden ISA-95-standardin määrittämä Equipment Level arvo on Enterprise. TUNI-Tampere universities -olion määrittämiseen käytetäänkin EnterpriseEquipmentClassType-määrittelyä. Muita EquipmentClassType-tyyppisiä Equipment Level -arvon määrittäviä tyyppimäärittelyitä OPC UA ISA-95-palvelimen osoiteavaruudessa ovat SiteEquipmentClassType, joka määrittää Site Equipment Level -arvon, AreaEquipmentClassType, joka määrittää Area Equipment Level -arvon ja ProcessCell-EquipmentClassType, joka määrittää Process Cell Equipment Level -arvon.

TUNI-Tampere universities -olio sisältää EquipmentLevel-muuttujan, jolla kuvataan olion Equipment level -arvo. Muuttujan lisäksi TUNI-Tampere universities -olioinstanssi koostuu TUNI - Hervanta Campus -nimisestä Equipment-instanssista. TUNI - Hervanta Campus -instanssin määrittelyyn käytetään SiteEquipmentClassType-tyyppistä tyyppimäärittelyä. TUNI - Hervanta Campus -olio sisältää Laboratory-olioinstanssin, joka puolestaan sisältää SimulatorCell-olioinstanssin. Sekä Laboratory-instanssiin että SimulatorCell-instanssiin liittyy EquipmentType-tyyppimäärittely. Laboratory-instanssi on lisäksi määritelty käyttäen AreaEquipmentClassType-tyyppimäärittelyä ja SimulatorCell-instanssi on määritelty käyttäen ProcessCellEquipmentClassType-tyyppimäärittelyä.

SimulatorCell-olio sisältää sellunkeittosimulaattoria kuvaavan BatchProcessSimulator -olion. BatchProcessSimulator-oliolle EquipmentClassType-määrittelyn tarjoaa SimulatorEquipmentClassType-olio. BatchProcessSimulator-olio koostuu Equipment-instansseista, joilla kuvataan simulaattoriin liittyviä laitteita. Simulaattorin eri laitteita kuvaavat Equipment-instanssit liittyvät BatchProcessSimulator-olioon MadeUpOfEquipment-viitteillä. OPC UA ISA-95-palvelimen osoiteavaruuteen on toteutettu jokaiselle erityyppiselle laitteelle oma EquipmentClassType-tyyppi. Esimerkkejä osoiteavaruudessa olevista prosessilaitteiston osia kuvaavista EquipmentClassType-tyyppisistä tyyppimäärittelysolmuista ovat muun muassa FlowIndicatorEquipmentClassType, jolla määritellään virtausmittari, sulkuventtiilien määrittelyyn käytettävä OnOffValveEquipmentClassType sekä ControlValveEquipmentClassType, jolla määritellään prosessilaitteiston säätöventtiilit.

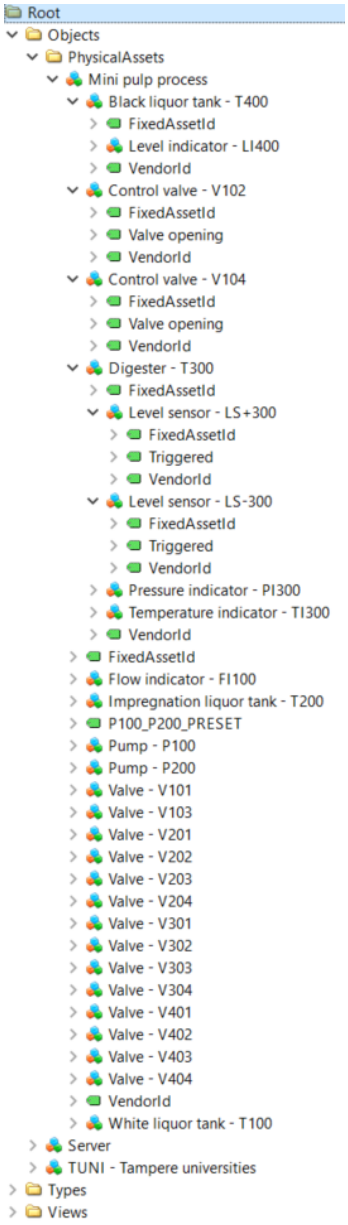
Sellunkeittosimulaattorin tietomallista poiketen OPC UA ISA-95-palvelinsovelluksen osoiteavaruuteen on sellunkeittosimulaattorin sisältämät tankit sekä niihin liittyvät anturit ja toimilaitteet mallinnettu laitekokonaisuuksiksi. Toisin sanoen prosessilaitteiston tankki on kooste siihen liittyvistä antureista ja toimilaitteista. Kuten BatchProcessSimulator-olion tapauksessa, tankin koostuminen muista laitteista kuvataan MadeUpOfEquipment-viitteellä.



Kuva 3.3. *UaExpert-asiakassovelluksen näkymä OPC UA ISA-95-palvelimen osoiteavaruuteen, jossa on kuvattu OPC UA ISA-95-liitännäisspesifikaation mukaisen palvelimen Equipment-hierarkia.*

OPC UA ISA-95-liitännäisspesifikaation mukaisen palvelimen osoiteavaruuteen määritetyt fyysisiä laitteita kuvaavat oliot on järjestetty FolderType-tyyppisen PhysicalAssets-olion alle. PhysicalAsset-olio sisältää Mini pulp process -nimisen olion, joka kuvaa fyysistä panosprosessisimulaattoria. Mini pulp process -olio koostuu muista fyysisen prosessin muodostavista laitteista. Mini pulp process -olio, sekä muut järjestelmän koosteiset PhysicalAsset-olioinstanssit, kasataan muista järjestelmän fyysisiä laitteita kuvaavista olioista käyttäen MadeUpOfPhysicalAsset-viitettä. Jokaiseen fyysistä laitetta kuvaavaan olioon liittyy PhysicalAssetClassType-määrittely. Jokaiselle erityyppiselle laitteelle on kuvattu omat PhysicalAssetClassType-määrittelyt sekä PhysicalAssetClassType-määrittelyihin kuuluvat attribuutit ja ominaisuudet. PhysicalAssetClassType-määrittely liittyy olioihin DefinedByPhysicalAssetClass-viitteellä. Fyysisiä laitteita kuvaaviin solmuihin liittyy HasTypeDefinition-viitteellä myös PhysicalAssetType-tyyppimäärittely. PhysicalAssetType-määrittelyyn liittyy sekä attribuutteja että ominaisuuksia. OPC UA ISA-95-palvelimen osoiteavaruudessa PhysicalAssetType-määrittelyyn liittyviä attribuutteja ovat FixedAssetId- sekä VendorId-attribuutit. Nämä attribuutit liittyvät kaikkiin osoiteavaruudessa

oleviin fyysisiä laitteita kuvaaviin solmuihin HasISA95Attribute-viitteellä. PhysicalAssetType-tyyppimäärittelyihin liittyviä ominaisuuksia on käytetty kuvaamaan tietyn laitteen tunnusomaisia piirteitä. Esimerkiksi sulkuventtiiliä kuvaavalla solmulla on Open-ominaisuus, jolla kuvataan sitä, onko venttiili avattu vai suljettu. Ominaisuudet liittyvät fyysisiä laitteita kuvaaviin olioihin HasISA95Property-viitteellä. Kuvassa 3.4 on esitetty joitakin OPC UA ISA-95-liitännäisspesifikaation mukaisen palvelimen fyysisiä laitteita kuvaavia olioita sekä niihin liittyviä ominaisuuksia.



Kuva 3.4. *UaExpert-asiakassovelluksen näkymä OPC UA ISA-95-palvelimen osoiteavaruuteen, jossa on esitetty palvelimeen liittyviä PhysicalAssetType-tyyppisiä olioita.*

Loogista laitetta kuvaavan osoiteavaruuden solmun liittymistä fyysistä laitetta kuvaavaan olioon kuvataan ImplementedBy-viitteellä. TUNI - Tampere universities -olion sisältämässä oliohierarkiassa BatchProcessSimulator-olion sisällä olevilla loogisilla laitteilla on ImplementedBy-viite fyysisiä laitteita kuvaaviin olioihin. Esimerkiksi loogista FI100-virtausmittaria kuvaava solmu viittaa ImplementedBy-viitteellä Flow indicator - FI100 -solmuun.

4 TULOKSET JA NIIDEN TARKASTELU

Opinnäytetyössä onnistuttiin toteuttamaan OPC UA ISA-95-liitännäisspesifikaation mukainen palvelin, joka muuttaa minipanosprosessia ohjaavan OPC UA -palvelimen osoiteavaruuden tiedon OPC UA ISA-95-liitännäisspesifikaation määrittämän informaatiomallin mukaiseksi. Palvelimen osoiteavaruuteen mallinnetun tiedon rakenne vastaa OPC Foundation -säätiön OPC UA ISA-95-liitännäisspesifikaatiossa määritettyä tiedon esitystapaa. Näin ollen voidaan todeta, että työn tavoitteet ovat täyttyneet.

Palvelimien välisen tiedonsiirron toimivuutta kokeiltiin käyttäen Unified Automation -yhtiön UaExpert-asiakassovellusta. Testauksen yhteydessä yksi asiakassovellus liittyi panosprosessia ohjaavaan OPC UA -palvelimeen ja toinen asiakassovellus liittyi opinnäytetyön yhteydessä toteutettuun OPC UA ISA-95-palvelimeen. OPC UA -asiakassovellusten avulla voitiin muuttaa palvelimien sisältämien muuttujien arvoja ja monitoroida muuttujissa tapahtuvia muutoksia. Testin perusteella voitiin havaita, että muutokset toisen palvelimen osoiteavaruuden sisältämissä muuttujissa saivat aikaan myös toisen palvelimen sisältämien muuttujien muutoksen.

Työn toteutuksen kannalta suurin haaste oli käytettävien ohjelmistokirjastojen heikko dokumentaatio. Node-opcua-isa95 -ohjelmistomoduulille ei varsinaista dokumentaatiota ole tarjolla, joten ohjelmistokomponentin käyttämiseksi moduulin lähdekoodia tuli tutkia melko perusteellisesti, mikä vaikeutti ja hidasti työn toteuttamista. Ohjelmistokirjastojen toiminnassa oli havaittavissa myös virheitä. Esimerkiksi node-opcua-isa95 -ohjelmistomoduulin lähdekooditiedoston perusteella PhysicalAssetClassType-määrittelyä luotaessa tyyppimäärittelylle voidaan antaa parametri, jolla määritetään laitteen valmistaja. Vaikka tyyppimäärittelyn luonnissa annetaan tieto laitteen valmistajasta, osoiteavaruuteen syntyneen tyyppimäärittelyyn liittyvä valmistaja on aina manufacturer.

Koska toteutetun OPC UA -palvelimen tietomalli on OPC UA ISA-95-informaatiomallin mukainen ja näin ollen yhteensopiva ISA-95-standardissa määritetyn tiedon esitystavan kanssa, voitaisiin palvelinta hyödyntää esimerkiksi tilanteessa, jossa panosprosessisimulaattori integroidaan tehtaan tuotannonohjaus- tai ERP-järjestelmän kanssa. Toteutetun palvelimen avulla panosprosessille voidaan siirtää ja panosprosessista voidaan lukea ISA-95-standardin tietomallien mukaisesti muotoiltua tietoa.

5 YHTEENVETO JA PÄÄTELMÄT

Opinnäytetyössä toteutettiin OPC Foundation -säätiön määrittämän OPC UA ISA-95-liitännäisspesifikaation mukainen palvelin. Palvelimen tehtävä on tarjota tapa, jolla sellunkeittoprosessia simuloivaa panosprosessilaitteistoa ohjaavan OPC UA -palvelimen osoiteavaruudessa olevaa tietoa voidaan käsitellä ISA-95-standardin määrittämien tietomallien avulla.

Ensimmäiseksi kandidaatintyössä esiteltiin keskeiset taustatiedot. Työn kannalta keskeisiä standardeja ja spesifikaatioita ovat OPC UA, ISA-95 sekä OPC UA ISA-95-liitännäisspesifikaatio. Opinnäytetyössä on esitelty keskeiset menetelmät, joilla eri standardien mukaiset tietomallit rakennetaan. Palvelimen toteutuksessa on esitetty keskeiset kohdat siitä, miten OPC UA ISA-95-liitännäisspesifikaation mukainen palvelin toteutetaan avoimen lähdekoodin OPC UA -ohjelmistokirjastolla. Työn toteutusvaiheessa on kuvattu myös, miten sellunkeittosimulaattoria ohjaavan OPC UA -palvelimen osoiteavaruuden tieto on mallinnettu OPC UA ISA-95-liitännäisspesifikaation mukaisen palvelimen osoiteavaruuteen.

Opinnäytetyöhön liittyvän OPC UA ISA-95-palvelimen osoiteavaruus perustuu sellunkeittoprosessia simuloivan prosessilaitteiston OPC UA -palvelimen osoiteavaruuteen. OPC UA ISA-95-palvelimen tietomalli on saatu aikaan muuttamalla prosessilaitteiston OPC UA -palvelimen osoiteavaruuden informaatio ensin ISA-95-standardissa määritettyjen tietomallien mukaiseksi ja tämän jälkeen muuttamalla se yhteensopivaksi OPC UA ISA-95-liitännäisspesifikaatiossa kuvattujen tietomallien kanssa.

OPC UA ISA-95-palvelin toteutettiin Node.Js -ympäristöön. Toteutuksessa hyödynnettiin node-opcua ja node-opcua-isa95 -ohjelmistopaketteja. OPC UA ISA-95-palvelimen toteutuksen perusteella voidaan todeta, että käytetyt ohjelmistopaketit soveltuvat hyvin OPC UA ISA-95-liitännäisspesifikaation mukaisen palvelimen toteuttamiseen heikosta dokumentaatiosta sekä ohjelmistokirjastojen yksittäisistä toimintavirheistä huolimatta.

Toteutetun OPC UA ISA-95-palvelimen liittäminen sellunkeittosimulaattorin OPC UA -palvelimeen on esimerkki siitä, miten tuotantoprosessi voidaan integroida tehtaan ylemmän hierarkiatason kanssa. OPC UA -protokolla on vahvasti käytössä teollisuudessa, joten sen hyödyntäminen tiedon siirtämiseen myös tehtaan eri kerrosten välillä on hyödyllistä. OPC UA ISA-95-liitännäisspesifikaation avulla OPC UA -standardin tarjoamat edut tiedonsiirtoprotokollana voidaan hyödyntää tehdastason eri kerrosten välillä, sillä liitännäisspesifikaatio määrittää tavan, jolla siirrettävän tiedon rakenne on eri tehdasker-

rosten välillä yhteensopivaa. Näin ollen tuotantoprosessin ohjaaminen esimerkiksi ERP-järjestelmästä onnistuisi käyttäen OPC UA -protokollaa ja ISA-95-standardin määrittämiä tietorakenteita.

LÄHTEET

[1] *ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod) Enterprise-Control System Integration Part 1: Models and Terminology*. 2010. ISBN: 9781936007479.

[2] *ANSI/ISA-95.00.02-2010 (IEC 62264-2 Mod) Enterprise-Control System Integration Part 2: Object Model Attributes*. 2010. ISBN: 9781936007486.

[3] *Business To Manufacturing Markup Language (B2MML). MESA International*. URL: http://www.mesa.org/en/B2MML.asp (viitattu 10. 09. 2019).

[4] *COMPASS. CEN*. URL: https://www.cen.eu/news/brochures/brochures/Compass.pdf (viitattu 10. 09. 2019).

[5] M. Damm, S.-H. Leitner & W. Mahnke. *OPC Unified Architecture*. English. 1. Aufl.; 1. Berlin, Heidelberg: Springer-Verlag, 2009. ISBN: 9783540688983. DOI: 10.1007/978-3-540-68899-0.

[6] T. Hannelius, M. Salmenperä & S. Kuikka. *Roadmap to adopting OPC UA*. English. 2008. DOI: 10.1109/INDIN.2008.4618203.

[7] *Harjoitustyö: Panosprosessin ohjaus, Automaation reaaliaikajärjestelmät, Automaation ja hydrauliikan laboratorio*.

[8] *Node.js. Node.js Foundation*. URL: https://nodejs.org (viitattu 10. 09. 2019).

[9] *OPC Unified Architecture Specification Part 1: Overview and Concepts*. 2017. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/.

[10] *OPC unified architecture Specification - Part 3: Address Space Model*. 2017. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-3-address-space-model/.

[11] *OPC Unified Architecture Specification - Part 5: Information Model*. 2015. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model/.

[12] E. Rossignon. *node-opcua*. URL: https://github.com/node-opcua/node-opcua (viitattu 10. 09. 2019).

[13] E. Rossignon. *node-opcua-isa95*. URL: https://github.com/node-opcua/node-opcua-isa95 (viitattu 10. 09. 2019).

[14] O. Toshio, A. Shahzad, H. Paul & B. Dennis. *OPC Foundation Companion Specification: OPC Unified Architecture for ISA-95 Common Object Model*. 2013.

[15] *What is a Standard? CEN*. URL: https://www.cen.eu/work/ENdev/whatisEN/Pages/default.aspx (viitattu 10. 09. 2019).